VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Síťová komunikace v počítačových hrách

Network communication in computer games

2011 **Bc. Libor Koláček**

Prohlášení:
Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Praze, dne 27. 4. 2011 …….

Poděkování
Velmi rád bych poděkoval a vyslovil uznání všem, kteří mi pomáhali při vzniku této práce. Především Ing. Janu Platošovi, vedoucímu mé diplomové práce, za trpělivé vedení a množství cenných rad a připomínek.

Abstrakt

U počítačových her je stále větší obliba hry pro více hráčů. Síťová komunikace u her však přináší řadu problémů, se kterými se musí programátor, tvořící síťovou hru, vypořádat. Cílem této diplomové práce je shrnout základní problémy při tvorbě multiplayeru a jejich možná řešení. Tato práce si neklade za cíl vyřešit všechny problémy a optimalizace spojené se síťovou komunikací, ale může posloužit jako návod, kde se vyvarovat některých základních chyb. Na výběr také přichází řada síťových knihoven, usnadňujících síťovou komunikaci. V následujícím textu tak budou některé z nich popsány a otestovány. Na jednoduché hře bude následné ukázán jeden z možných přístupů k síťové komunikaci, a budou na ní vysvětleny základní principy a možné problémy, se kterými se tvůrce hry pro více hráčů může setkat.

Klíčová slova

C++, DirectPlay, Síť, Multiplayer, Hry

Abstract

The world of computer games has been changing rapidly in the favour of multiplayer games recently. This change has brought new challenges to programmers who now have to solve many issues the network connection causes. The goal of this thesis is to summarize basic issues which often occur during creating multiplayer solutions. Of course, this paper doesn’t aspire to solve all possible connection and optimization issues. Though it could be a guide to help others avoiding basic mistakes. It also deals with a vast range of network libraries. Some of them are described in this text, together with their test results. With using an example of a simple game, this text also describes one of many possible solutions of creating a network communication which a developer of multiplayer games may encounter.

Keywords

C++, DirectPlay, Network, Multiplayer, Games

Seznam použitých zkratek a symbolů

JSON – JavaScript Object Notation
UDP – User Datagram Protocol
TCP – Transmission Control Protocol
NAT – Network Address Translation
FPS – first person shooter
MMO – Massive Multiplayer Online
OpenGL – Open Graphics Library
GLUT – OpenGL Utility Toolkit
OS – Operační systém

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

Multiplayer u počítačových her je dnes stále rozšířenější. Vytvořit kvalitní hru pro více hráčů však není zcela triviální a přináší s sebou řadu specifických problémů. Ty nastávají převážně v synchronizaci mezi jednotlivými klienty a serverem. Tato diplomová práce se zabývá různými způsoby síťové komunikace u počítačových her a s tím spojených problémů. Dále rozvádí některá možná řešení a způsoby optimalizace. Nepojednává o všech možných problémech, ale zabývá se pouze těmi základními.

Technologie potřebné pro samotnou síťovou komunikaci by měl znát každý, kdo by chtěl pracovat na multiplayerové hře. Jedná se především o transportní protokoly TCP a UPD, používaných při síťové komunikaci. Není důležité však znát tyto protokoly do detailů, ale je výhodné ovládat alespoň princip a jejich základy. O veškeré detaily se totiž starají síťové knihovny, tudíž se programátor nemusí starat o nižší úrovně komunikace. Je ale dobré porovnat alespoň základní výhody a nevýhody a jejich vhodnost pro využití u počítačových her. Této problematice je věnována samostatně druhá kapitola.

Jedním z důležitých momentů při tvorbě hry více hráčů může být samotný výběr síťové knihovny. Tato práce se bude zabývat zejména knihovnami použitelných v jazyce C++, kterých je dnes k dispozici poměrně velké množství. Při rozhodování je potřeba vycházet z několika vlastností, které určují kvalitní produkt. Kromě základních vlastností je hlavní předností každé knihovny její výkon a dostatečně rychlé zpracování požadavků od klienta a zpětná odezva. Srovnáním základních vlastností jednotlivých knihoven a zátěžovými testy se zabývá třetí a čtvrtá kapitola.

Pátá kapitola se zabývá samotnou síťovou komunikací a problémům vzniklých při hře více hráčů. Jsou zde podrobněji rozepsány některé možné způsoby řešení komunikace mezi klientem a serverem a způsoby synchronizace. Jsou zde popsány některé základní principy, kterými se dá vytvořit síťová hra pro více hráčů. Dále jsou zde nastíněny některé problémy, způsobené zpožděním při přenosu dat, a jejich možná řešení.

Poslední část tohoto textu je věnována popisu implementace jedné jednoduché síťové hry. Jejím účelem není dokonalá hratelnost, ale ukázat na příkladech některé z problémů, které mohou vzniknout při tvorbě tohoto typu her. Budou tak zdokumentovány a popsány vzniklé potíže z praxe, v závislosti na předchozí teorii.

2 Síťová komunikace

Tato kapitola bude zaměřená na síťovou komunikaci použitou v počítačových hrách. Převážně pak na transportní protokoly TCP a UDP. Účelem není detailně rozebírat komunikaci po síti, ale shrnout pouze jejich základy pro větší přehled.

2.1.1 Protokoly TCP a UDP

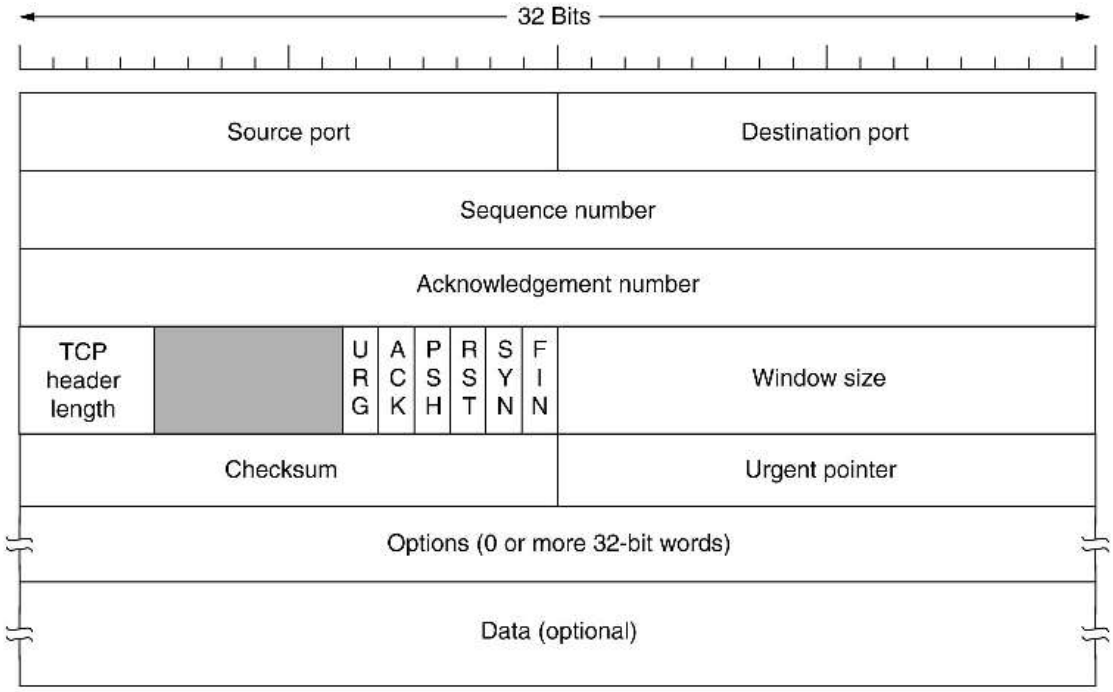
Pro komunikaci po síti se používají transportní protokoly TCP nebo UDP. Jedná se o protokoly z rodiny ISO/OSI, které se starají o odesílání datových balíčků (packetů). Mezi těmito dvěma protokoly je zásadní rozdíl v přístupu navázání spojení s příjemcem dat a následnou kontrolou odeslaných dat. Tyto dva protokoly jsou popsány v následujících podkapitolách.

2.1.1.1 Protokol TCP

Tento protokol využívá tzv. *handshaking*. Jedná se o obousměrnou komunikaci mezi dvěma síťovými subjekty. V podstatě se jedná o kontrolu připojení a ověřování odeslaných dat. To přináší řadu výhod, jako je spolehlivost přenosu (kontrola doručení datového packetu, zachování pořadí odeslaných dat a další).

Nevýhodou však je, že tato metoda vyžaduje podstatně vyšší režie pro přenos dat. Stává se tak poměrně nevýhodnou v případě, kdy potřebujeme rychlou komunikaci. Pokud je však upřednostňován výkon před spolehlivostí, je vhodnější protokol UDP (viz. kapitola 2.1.1.2).

Na následujícím obrázku je vyobrazena hlavička protokolu TCP.



Obrázek I: Hlavička protokolu TCP [7]

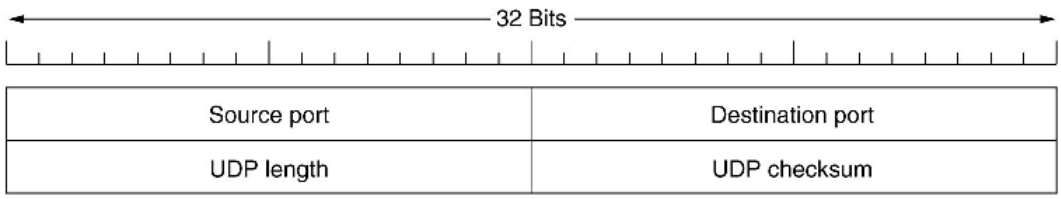
Ta obsahuje identifikaci příjemce i odesílatele. Navíc v sobě obsahuje kromě samotných dat také informace, sloužící pro handshaking a další kontroly. Obaluje tak samotná tak o další informace, které je nutné odeslat po síti, což zapříčiňuje vyšší nároky na datový tok.

2.1.1.2 Protokol UDP

Tento protokol je na rozdíl od TCP jednodušší. Ze síťového zdroje se odesílají nezávislá data, u kterých neprobíhá kontrola doručení. V některých případech může docházet ke ztrátě dat. Výhodou jsou však velmi nízké režie a tudíž i vyšší výkon.

Z těchto důvodů jsou síťové knihovny využívané v počítačových hrách postaveny převážně na tomto protokolu. Případná ztráta dat je však potřeba ošetřit.

Na následujícím obrázku je vyobrazena hlavička protokolu UDP.



Obrázek II: Hlavička protokolu UDP [7]

Na rozdíl od hlavičky TCP obsahuje pouze identifikaci příjemce a odesílatele a kontrolní součet a velikosti dat. Minimalizují se tak náklady na velikost odesílaných dat.

3 Síťové knihovny

Následující kapitola se zabývá síťovými knihovnami v jazyce C++ a jejich srovnáním. Usnadní tak rozhodování při volbě správné knihovny. V dnešní době je k dispozici řada možností, ze kterých je možné se rozhodnout. Budeme se však zabývat pouze volně dostupnými knihovnami (pro nekomerční použití), které jsou většinou dostupné také se zdrojovými kódy (vydány jako open source).

3.1 Popis síťových knihoven

Nyní se podíváme na přehled testovaných knihoven. V následující stručné tabulce jsou shrnuty všechny knihovny, které budou otestovány, a jejich základní informace.

Název	Open source	Licence	Protokol
DirectPlay	Ne	Součástí DirectX 8	UDP
RakNet	Ano	Zdarma pro nekomerční použití	UDP
OpenTNL	Ano	Zdarma pro nekomerční použití	UDP
HawkNL	Ano	Zdarma	UDP
ENet	Ano	Zdarma	UDP
Zoidcom	Ne	Zdarma pro nekomerční použití	UDP
SyncSys	Ano	Zdarma (Součástí zlib)	UDP

Tabulka I: Seznam síťových knihoven

Nejedná se samozřejmě o dnes všechny dostupné knihovny, ale je to základní výčet těch nejpoužívanějších knihoven na trhu. Čtenář si tak může udělat přehled a ulehčit si tak rozhodování.

Všechny tyto knihovny jsou postavené nad protokolem UDP. Některé knihovny však poskytují mechanismy kontroly odeslaných UDP packetů v jejich pořadí. To může usnadnit vývojáři práci, neboť se již nemusí starat o kontrolu odeslaných a přijatých packetů. Spojují se tak společné výhody protokolů UDP (rychlost) a TCP (ověřování odeslaných dat).

3.1.1 DirectPlay

DirectPlay je součástí knihovny DirectX 8, která je od srpna 2007 označena jako zastaralá. Naposledy se vyskytovala jako součást již zmíněných DirectX 8.1, ale u novějších verzí DirectX (DirectX 9 a novější) se již síťová podpora nevyskytuje. Nahrazují je Windows Sockets 2 (WinSock), které jsou Microsoftem doporučovány jako vhodnější náhrada.

Z těchto důvodů by si měl programátor rozmyslet, zda má vyvíjet v zastaralé, ač poměrně kvalitní, knihovně, nebo zvolit některou z kvalitnějších a novějších knihoven.

Ať už se jedná o klasické připojení klient-server nebo peer-to-peer, DirectPlay vždy běží v samostatném vlákně, které se stará o odesílání a přijímání dat. Podpora těchto vícevláknových aplikací tak zvyšuje výkon při práci v síti.

3.1.1.1 Výhody

Ačkoliv se jedná o již zastaralou knihovnu, která se v dnešní době nedoporučuje používat, jedná se o poměrně kvalitní knihovnu s velkým množstvím možností. Nechybí podpora Peer-to-Peer

připojení, základního vyhledávání v lobby a mnoho dalších funkcí. Někteří vývojáři, fandící firmě Microsoft, si mohou mezi výhody připočíst také to, že se jedná o produkt firmy Microsoftu. Tato výhoda však pro některé může být i nevýhodou.

V neposlední řadě stojí za zmínku poměrně kvalitně zpracované ukázkové příklady, které jsou součástí DirectX 8 SDK.

3.1.1.2 Nevýhody

Mezi hlavní nevýhodu této knihovny patří především to, že již není několik let podporována. S touto podporou se tak vytrácí jakákoliv šance najít nějaký kvalitní zdroj informací, dokumentace a tutoriály, které by usnadnily vývoj s tímto nástrojem.

3.1.2 RakNet

RakNet je jedna z kvalitních open source síťových knihoven, která je poskytována nezávislým vývojářům pro nekomerční použití zcela zdarma. Pro komerční použití je dostupná za poplatek. K dnešnímu dni je již vydána verze 4, a stále funguje plná podpora ze strany vývojářů.

RakNet také poskytuje funkce vyšších úrovní, jako je přenos zvuku, vyhledávání serveru v lobby, zabezpečení síťové komunikace, podpora NAT, a další. Podporuje také již zminou kontrolu doručených packetů v takovém pořadí, v jakém byli data odeslány [1].

Za zmínku také může stát herní engine Unity, který tuto knihovnu používá pro síťovou komunikaci. Také je s oblibou využívána i u některých komerčních aplikací a her.

3.1.2.1 Výhody

Velkou výhodou může být pro někoho podpora více platforem, tedy nejen PC. S touto knihovnou lze vyvíjet aplikace na herní konzole Xbox360 a PS3. Mezi podporované operační systémy patří také Linux, Mac OS, iOS pro iPhone a další. Potěší také podpora Steamworks pro herní platformu Steam od společnosti Valve [1].

Hlavní její výhodou je však její profesionální poddání a implementace. Je velmi přehledná a jednoduchá na používání. Má také kvalitní dokumentaci a přehledné ukázkové příklady. Nemalou výhodou je také uživatelská podpora v podobě aktivního fóra.

3.1.2.2 Nevýhody

Jedinou nevýhodou může byt to, že pro komerční produkty již není zdarma. Dále je také nutná registrace i před stažením volné verze pro nekomerční použití. Toto jsou však pouze drobné detaily na jinak velmi kvalitně zpracované knihovně.

3.1.3 OpenTNL

Další zkoumanou knihovnou je německá open source síťová knihovna OpenTNL (Torque Network Library). Ta je poskytována společně se zdrojovými kódy pro neomezené použití u nekomerčních projektů. Jedná se o jednoduchou knihovnu, která poskytuje základní síťové funkce, ale nepřináší žádné výhody.

OpenTNL je součástí poměrně neznámého herního engine Torque Game Engine. Tuto knihovnu využívá ale například i herní engine Ogre3D, nad kterým je postavena jeho síťová komunikace.

3.1.3.1 Výhody

Tato knihovna nepřináší oproti ostatním mnoho výhod. Jednou z mála může být nezávislost na platformě, tudíž ji lze použít jak pod OS Windows, tak pod Mac OS a Linuxem [2].

3.1.3.2 Nevýhody

První nevýhoda, která může odradit mnoho vývojářů od použití této knihovny je nepřehledná webová prezentace na stránkách. Některé texty jsou stále v němčině a stáhnout tuto knihovnu proto může být pro někoho problém. Podaří-li se však knihovnu stáhnout, čeká na vývojáře již klasicky anglická dokumentace, která však bohužel nekoresponduje s nejaktuálnější verzí. Dá se však využít některých tutoriálů u již zmíněného enginu Ogre, kde se uživatel dočká také větší podpory.

3.1.4 HawkNL

Knihovna HawkNL je součástí Hawk Software (*HawkNL, HawkNLU, and HawkVoice*) a jejím autorem je Phil Frisbie, Jr, který ji vytvořil v roce 2003. Jedná se tedy o jednoduchou knihovnu, která poskytuje základní síťovou komunikaci pod protokolem UDP. Na rozdíl od předchozích knihoven neobsahuje základní kontrolu UDP packetů. Poslední úprava do finální podoby byla vytvořena v roce 2004, a od té doby již tato knihovna není podporována [3].

3.1.4.1 Výhody

Mezi výhody patří možnost využití i u komerčních aplikací a vlastní implementaci vícevláknového využití.

3.1.4.2 Nevýhody

Mezi hlavní nevýhody patří absence tutoriálu a nekvalitní dokumentace, ve které chybí některé základní prvky. V dnešní době poskytuje pouze základní funkčnost oproti ostatním knihovnám. Vzhledem k tomu, že se již několik let neposkytuje jakákoliv podpora k této knihovně, dá se považovat za zastaralou.

3.1.5 ENet

ENet je volně dostupná open source knihovna, kterou je možné použít i pro komerční využití bezplatně. Původně byla vyvinuta pro FPS hru Cube. Poté však vývojáři poskytli ENet volně ke stažení veřejnosti. ENet poskytuje základní síťovou komunikaci pod protokolem UDP, která obsahuje kontrolu odeslaných packetů a pořadí, v jakém byla odeslána [4].

3.1.5.1 Výhody

Kromě toho, že je volně dostupná i pro komerční použití obsahuje tato knihovna také velmi povedený tutoriál a dokumentaci. Je také velmi jednoduchá na pochopení a je spolehlivá. Její jednoduchost však může být brána též jako menší nevýhoda, pokud chce vývojář využívat některé rozšiřující funkce. U jednoduchých aplikací jsou ovšem tyto funkce irelevantní.

3.1.5.2 Nevýhody

Nevýhodou může být celková jednoduchost. Neposkytuje rozšiřující vyšší funkce jako např. RakNet a proto pro komplexnější úlohy může být leckdy nevyhovující. Nejedná se však o nijak zásadní nedostatek.

3.1.6 Zoidcom

Zoidcom je další z knihoven, která je volně dostupná pro nekomerční využití. Jako jedna z mála však neposkytuje své zdrojové kódy volně k dispozici. Zoidcom obsahuje kontrolu odeslaných packetů v pořadí, v jakém byly data odeslány. Navíc také přidává podporu NAT, kódování dat, možnost odesílání větších souborů na pozadí a další [5].

3.1.6.1 Výhody

Velkou výhodou je velmi snadná použitelnost a přehlednost nabízeného kódu. Dokumentace a přiložené příklady jsou zpracovány profesionálně a přehledně. Tato knihovna je tedy vhodná pro jednoduché síťové aplikace, neboť nabízí velmi pohodlný přenos dat, který pro mnohé aplikace bohatě dostačuje.

3.1.6.2 Nevýhody

Jednou z mála nevýhod může být nedostupnost zdrojových kódů. Záleží však na každém vývojáři, vidí-li to jako nevýhodu.

3.1.7 SyncSys

Poslední zkoumanou knihovnou je SyncSys, která je součástí rozsáhlejšího projektu „zlib“. Ta je opět poskytována jako open source, a je volně dostupná i pro komerční aplikace bez poplatků. Je zaměřená především na velké množství klientů připojené k jednomu serveru a je uzpůsobena na tvorbu serverů u MMO herních titulů (Massive Multiplayer Online). Server je optimalizován pomocí několika vláken. Tím je poskytnut vysoký výkon při zpracování velkého množství požadavků [6].

3.1.7.1 Výhody

Výhodou je volná licence i pro komerční použití a optimalizace serveru pro přijímání velkého množství požadavků. K dispozici je také zdrojový kód a řada příkladů. Jedná se tedy o poměrně kvalitní knihovnu, která poskytuje solidní síťovou komunikaci. Součástí je také jednoduchá dokumentace a řada ukázkových příkladů

3.1.7.2 Nevýhody

Mezi drobné výtky může patřit nepříliš povedená dokumentace a také chybějící tutoriál. Jedná se však o drobné prohřešky, které jsou zastíněny jejími výhodami.

3.2 Souhrnný přehled

Na závěr jsou shrnuty podporované funkce jednotlivých knihoven v přehledové tabulce. Po vykonání zátěžových testů je nad těmito knihovnami vynesený závěr v kapitole 4.4.

Název	Open source	Peer-to-peer	Přenos zvuku	Podpora NAT	Kontrola packetů	Multi-platformní
DirectPlay	✗	✓	✓	✓	✗	✗
RakNet	✓	✓	✓	✓	✓	✓
OpenTNL	✓	✗	✗	✗	✓	✓
HawkNL	✓	✗	✓	✗	✗	✓
ENet	✓	✗	✗	✗	✓	✓
Zoidcom	✗	✗	✗	✓	✓	✓
SyncSys	✓	✗	✗	✗	✗	✓

Tabulka II: Podporované funkce knihoven

Poskytování zdrojových kódů (open source) k dané knihovně již bylo popsáno v předchozí části této kapitoly. Následuje podpora peer-to-peer spojení. Tu překvapivě podporují pouze DirectPlay a RakNet. Přenos zvukového signálu v reálném čase je podporován také těmito dvěma knihovnami. Dále ho podporuje HawkNL. Tato funkce může být u počítačových her poměrně užitečná.

Podpora NAT (Network Address Translation) je podporována již tradičně prvními dvěma knihovny, dále ji však podporuje také Zoidcom. Tato podpora překladu adres může být při síťové komunikaci výhodná.

Kontrolu odeslaných packetů podporují knihovny RakNet, ENet, OpenTNL a Zoidcom. Jedná se o kontrolní mechanismus doručených dat ve správném pořadí. V případě ztrátě dat se knihovna postará o znovu odeslání ztracených balíčků. Jedná se o poměrně užitečnou funkci. Jak se však později ukáže, má tato vlastnost poměrně negativní vliv na rychlost přenosu dat.

Poslední z uvedených vlastností v tabulce je podpora různých operačních systémů. Téměř všechny knihovny lze zprovoznit na OS Windows/Linux/Unix, některé též na Mac OS. Výjimkou je pouze knihovna DirectPlay, jelikož je součástí knihovny DirectX, která je podporována pouze pod OS Windows. Knihovna RakNet navíc podporuje mobilní platformy a herní konzole, což je velké plus.

4 Zátěžové testy

Co nás však hlavně zajímá u síťových knihoven je jejich výkon. Ten je hlavní rozhodovací hodnotou. Testy proběhly dva. První byl prováděn na jednom počítači (localhost). Druhý na virtuální síti Hamachi na dvou fyzických počítačích mezi Ostravou a Prahou. V následujících podkapitolách budou popsány grafy s dobou odezvy pro jednotlivé knihovny.

V rámci testů proběhli také testy síťového rozhraní Windows Sockets 2 (zkráceně WinSock), které nejsou uvedeny mezi knihovnami. Jedná se totiž síťové rozhraní, které je součástí knihoven Windows. Pro lepší srovnání jsou proto zahrnuty mezi testy knihoven.

Každý test se skládal z odeslání 100 požadavků na server a čekání na jeho odpověď. V tomto čase se měřila odezva serveru. Každý test byl proveden několikrát, aby se zamezilo případné odchylce v měření a výsledky tak přinesly co nejpřesnější srovnání.

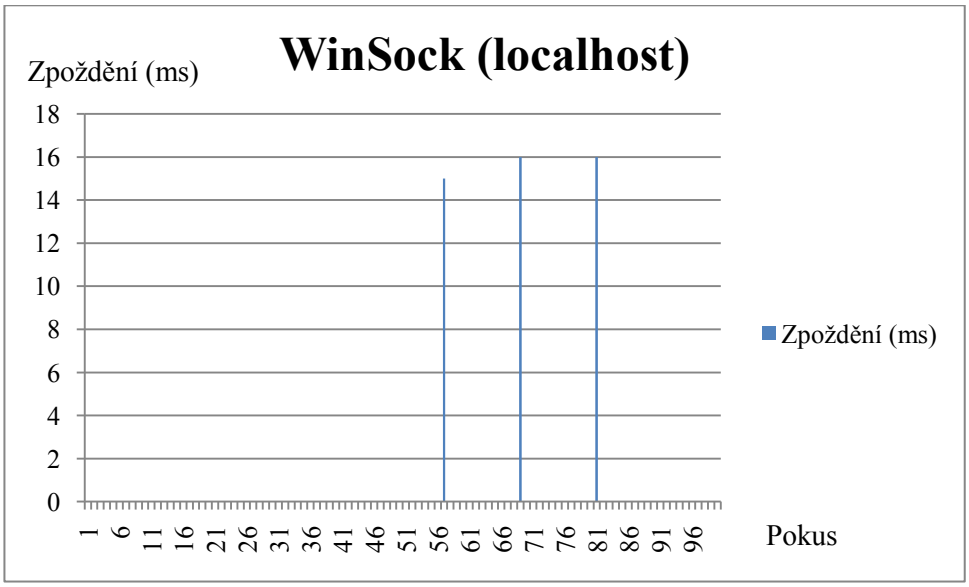
4.1 Localhost

V rámci testů jednotlivých knihoven proběhli nejprve testy na jednom počítači (localhost). Již zde byli patrné rozdíly.

Následující grafy popisují každou knihovnu zvlášť. Ke každé knihovně je přiřazen jeden graf, který zobrazuje zpoždění v každém pokusu (v každém testu proběhlo vždy 100 pokusů). Na závěr pak budou výsledky zprůměrovány a vykresleny v jednom přehledném grafu.

4.1.1 WinSock

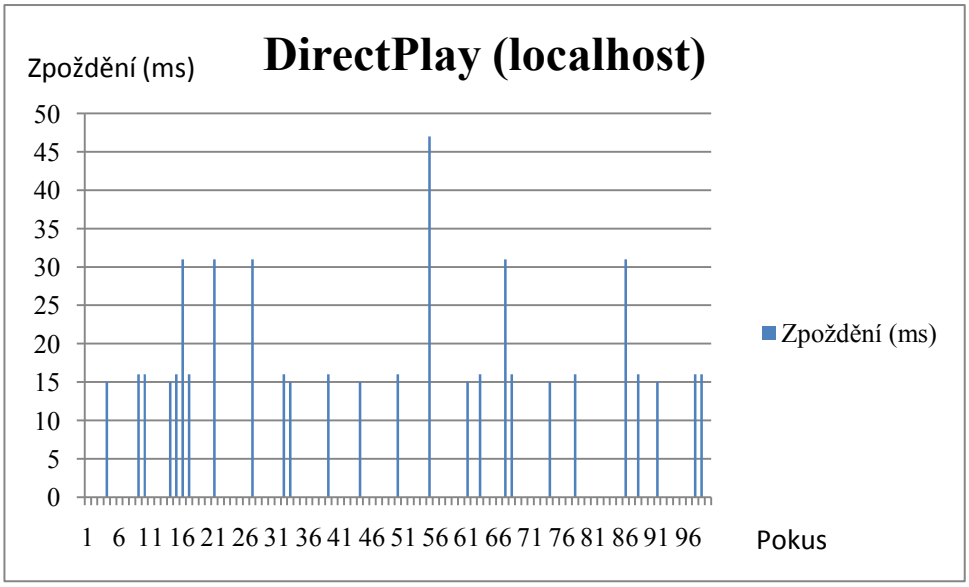
Knihovna WinSock dosáhla v testech na lokálním počítači velmi dobrých výsledků. Jak ukazuje následující graf, došlo pouze ke třem výkyvům, jinak byla ustálená velikost zpoždění 0ms. Tato odchylka je však zanedbatelná.



Graf I: WinSock (localhost)

4.1.2 DirectPlay

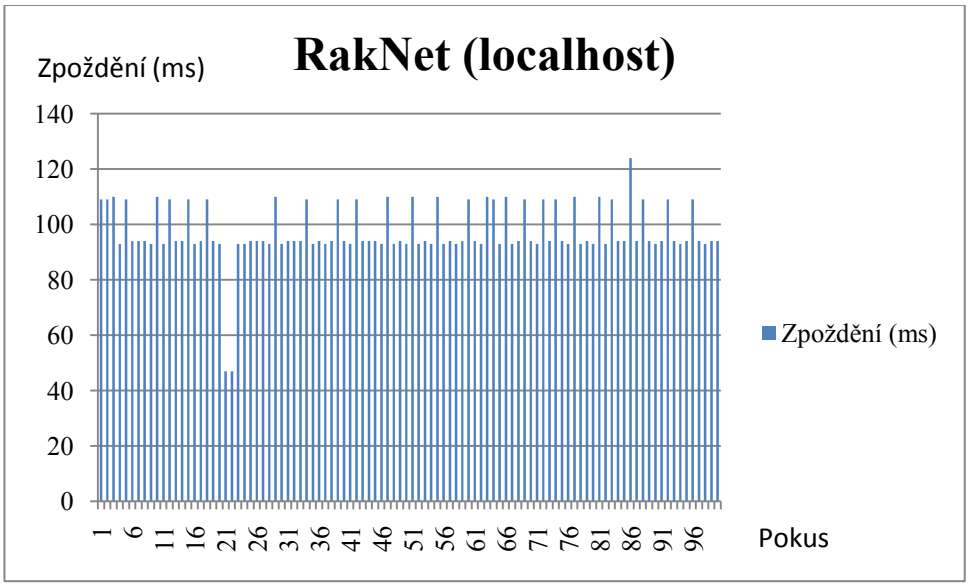
Knihovna DirectPlay již tak dobrých výsledků nedosahovala. V jednom případě dosáhla odezva téměř k 50ms. Stále však ve většině případů byla odezva 0ms. Přesto však občasné výkyvy k 15 nebo dokonce 30ms nelze přehlédnout.



Graf II: DirectPlay (localhost)

4.1.3 RakNet

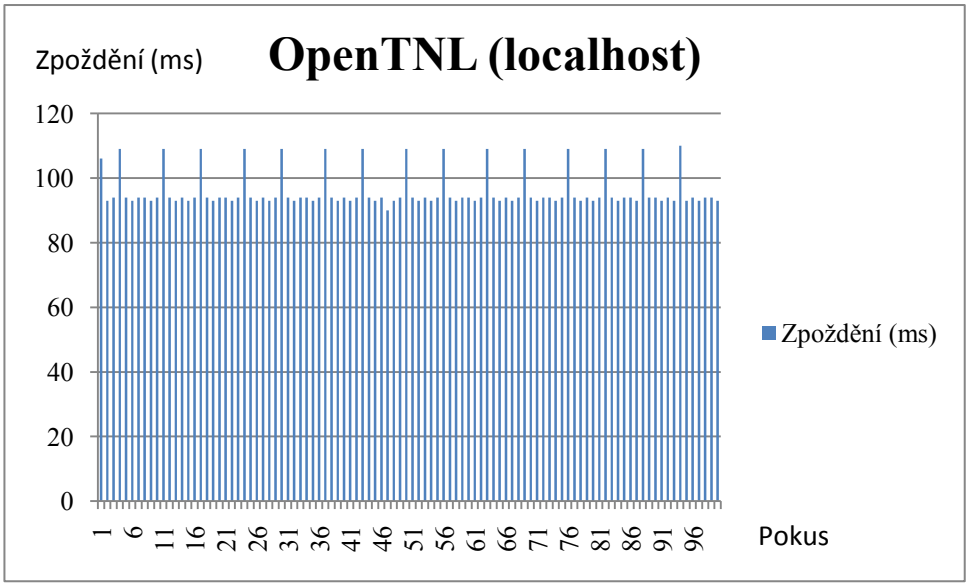
RakNet v testech na lokálním počítači neuspěla. Zpoždění se u všech pokusů pohybovalo okolo 100ms. Jedná se o poměrně velké zklamání, neboť RakNet byla jedna z nejlépe zpracovaných testovaných knihoven. Zpoždění zřejmě zapříčiňuje testování doručených packetů, což vede k vysokým latencím.



Graf III: RakNet (localhost)

4.1.4 OpenTNL

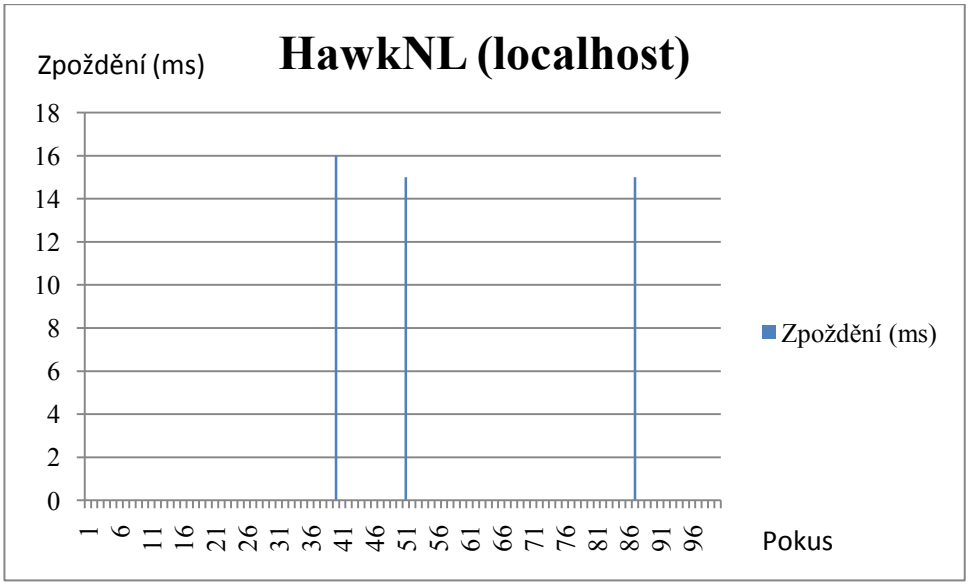
Stejně jako knihovna RakNet, i OpenTNL dosahovalo velkých zpoždění okolo 100ms na každý pokus. Zde se jedná pravděpodobně o stejný problém, neboť kontrola doručených UDP packetů může způsobovat takto vysoké latence.



Graf IV: OpenTNL (localhost)

4.1.5 HawkNL

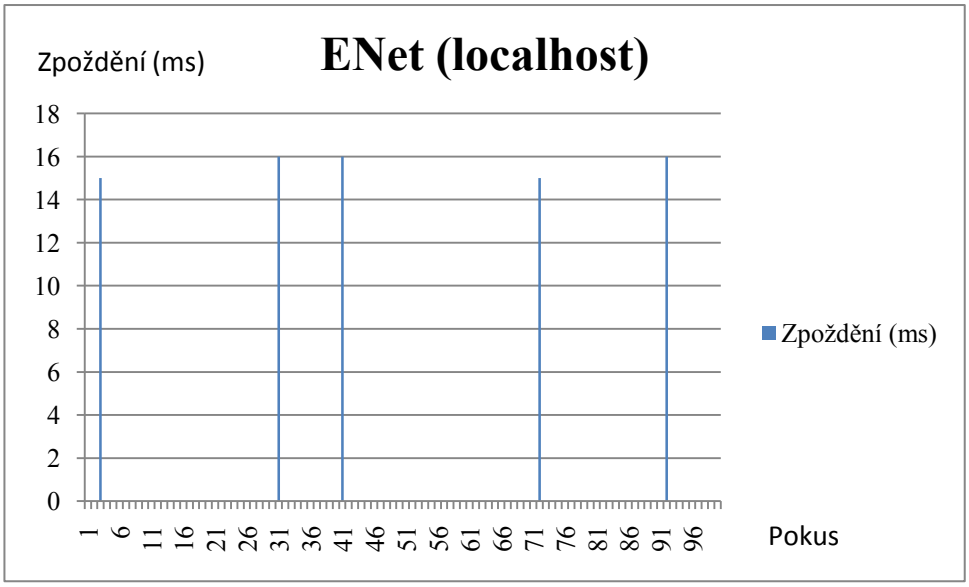
Knihovna HawkNL dosáhla velmi dobrých výsledků. Její odezva na localhost byla až na několik výjimek vždy 0ms. Tyto odchylky lze však považovat za irelevantní. Dosáhla tedy stejných výsledků jako v případě použití WinSock.



Graf V: HawkNL (localhost)

4.1.6 ENet

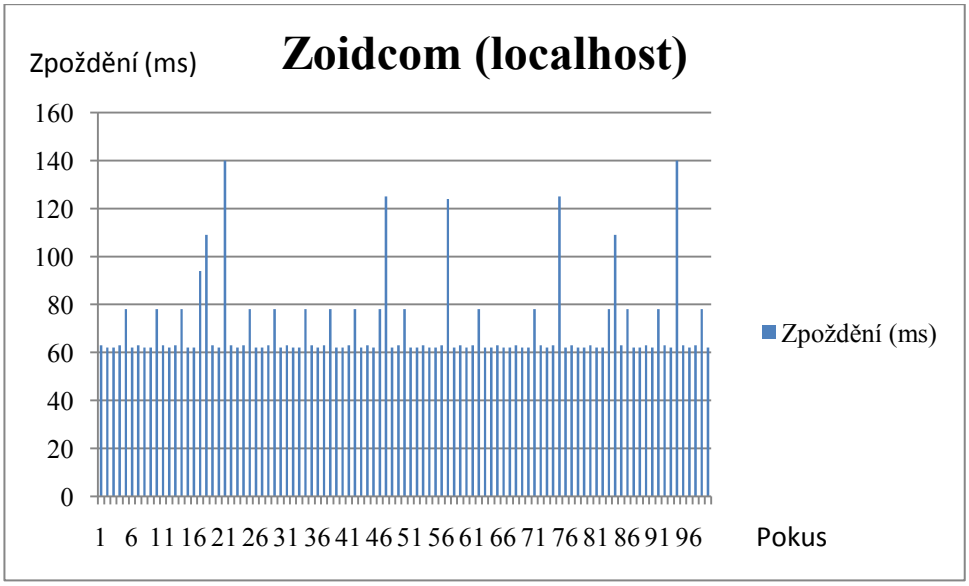
Stejně jako předchozí knihovna nebo WinSock, i ENet dosáhl velmi dobrých výsledků při testech na lokálním počítači. Odezva ze serveru byla téměř vždy 0ms, a stejně jako u předchozích testů lze považovat výjimky za nepodstatné.



Graf VI: ENet (localhost)

4.1.7 Zoidcom

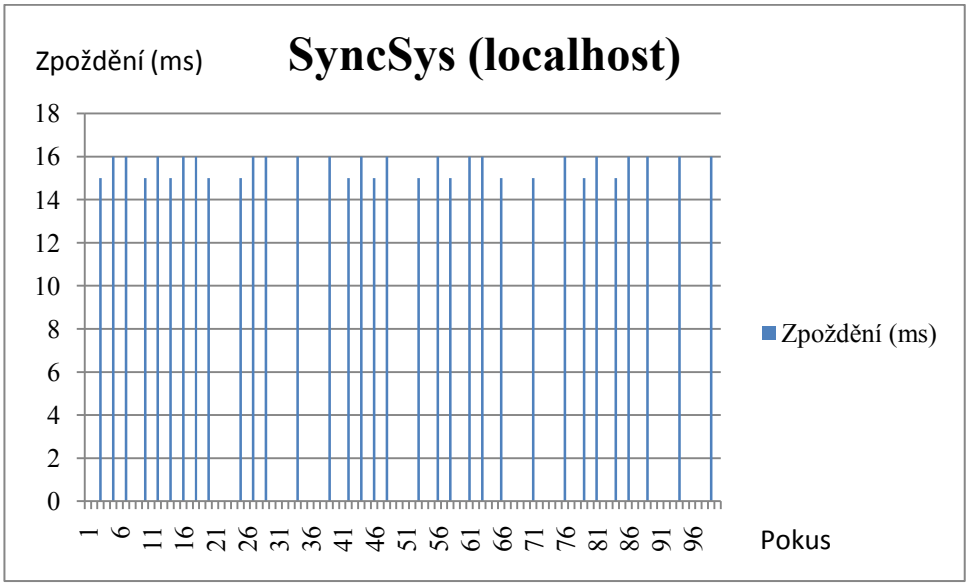
Knihovna Zoidcom již tak dobrých výsledků nedosahovala. Její průměrná latence činila ustálených 60ms. Nejedná sice o tak velké zpoždění jako v případě knihovny RakNet nebo OpenTNL, přesto je to však nezanedbatelné zpoždění.



Graf VII: Zoidcom (localhost)

4.1.8 SyncSys

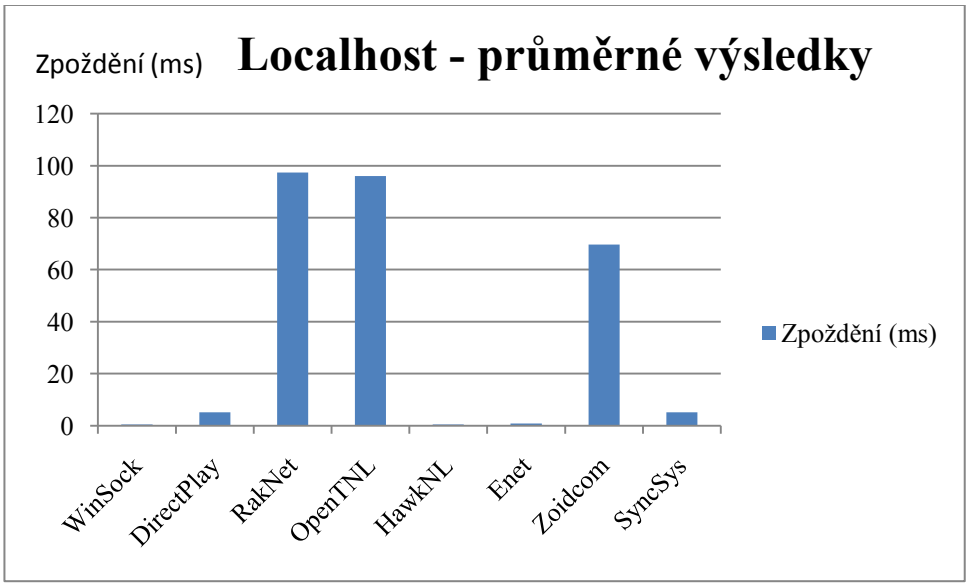
Knihovna SyncSys dosáhla v těchto testech poměrně dobrých výsledků. Přesto se jí nedají upřít občasné výkyvy latence. Tyto výkyvy však nikdy nepřesáhly čas 16ms, a proto se jedná o velmi uspokojivý výkon. Zařazuje se tak mezi ty knihovny, které dosáhli na lokálním počítači očekávané latence kolem 0ms.



Graf VIII: SyncSys (localhost)

4.1.9 Průměrné výsledky

Nyní přichází na řadu průměrování. Z každého testu je vytažen průměr ze sta pokusů, který je zobrazen v přehledovém grafu pro všechny knihovny. Dají se tak porovnat výsledky testů všech knihoven v jednom přehledném grafu.



Graf IX: Localhost – průměrné výsledky

Z konečných výsledků vyplývá, že knihovny WinSock, HawkNL a ENet dosáhli výsledků velmi uspokojivých. Jejich průměrná doba zpoždění nepřesáhla 1ms. V případě knihoven DirectPlay a SyncSys se již nejedná o úplně čistý výsledek, přesto je však lze považovat také za poměrně uspokojivé. Nejhůře v těchto testech obsáhli knihovny RakNet, OpenTNL a Zoidcom. Jejich latence na lokálním počítači dosahovala velmi špatných výsledků. V případě knihovny RakNet se jedná o poměrně nemilé překvapení.

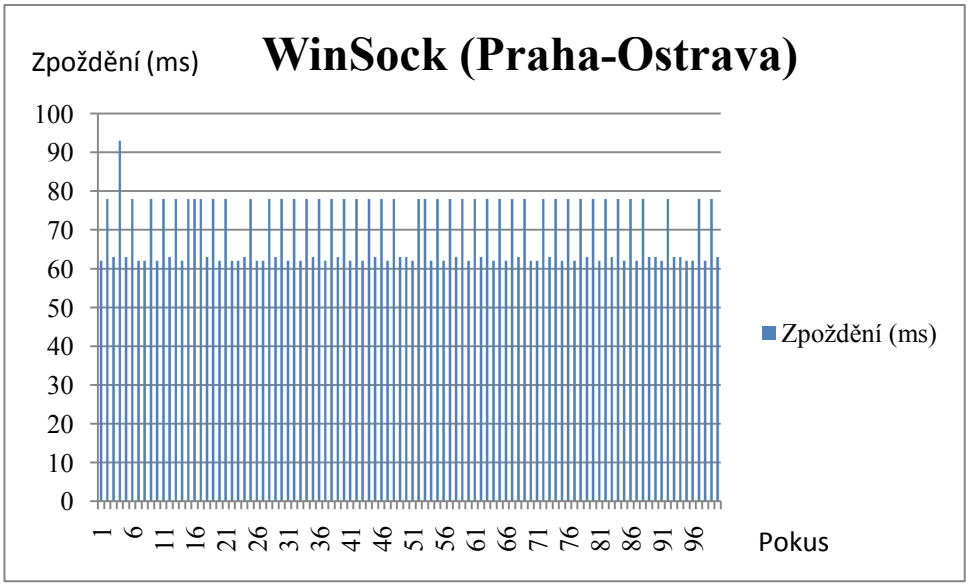
4.2 Virtuální síť mezi Prahou a Ostravou

Další testy proběhly na virtuální sítí Hamachi (konkrétně na verzi Hamachi 2 ve virtuální sítí LogMeIn [9]) mezi dvěma fyzickými počítači umístěnými v Praze a Ostravě. Při testech byla naměřena ping hodnota v průměru mezi 67-70ms. U takto velké vzdálenosti a při použití virtuální sítě může dojít k různým výkyvům ve spojení. Hodnota ping však byla během testů průběžně testována. Nebyly zaznamenány žádné významné výkyvy, tudíž spojení, při kterém testy probíhaly, lze považovat za stabilní.

Výsledky testů byly prováděny se stejnými postupy jako v případě na lokálním počítači. Jedná se tedy vždy i sadu 100 pokusů, které jsou zobrazeny v následujících grafech.

4.2.1 WinSock

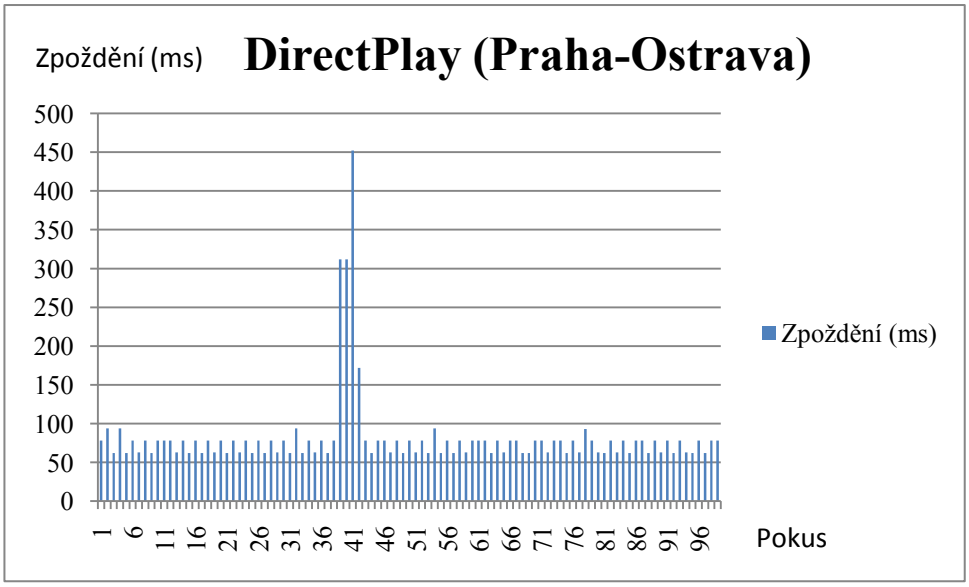
Stejně jako v případě testů na lokálním počítači, i zde dosahovala knihovna WinSock velmi dobrých výsledků. Odezva při testech dosahovala většinou úrovně velikosti ping hodnoty. Obvyklá rychlost odezvy byla pod 70ms a jen v některých případech se přiblížila k 80ms. Jedná se tak o nejlepší výsledky v porovnání s ostatními testovanými knihovnami.



Graf X: WinSock (Praha-Ostrava)

4.2.2 DirectPlay

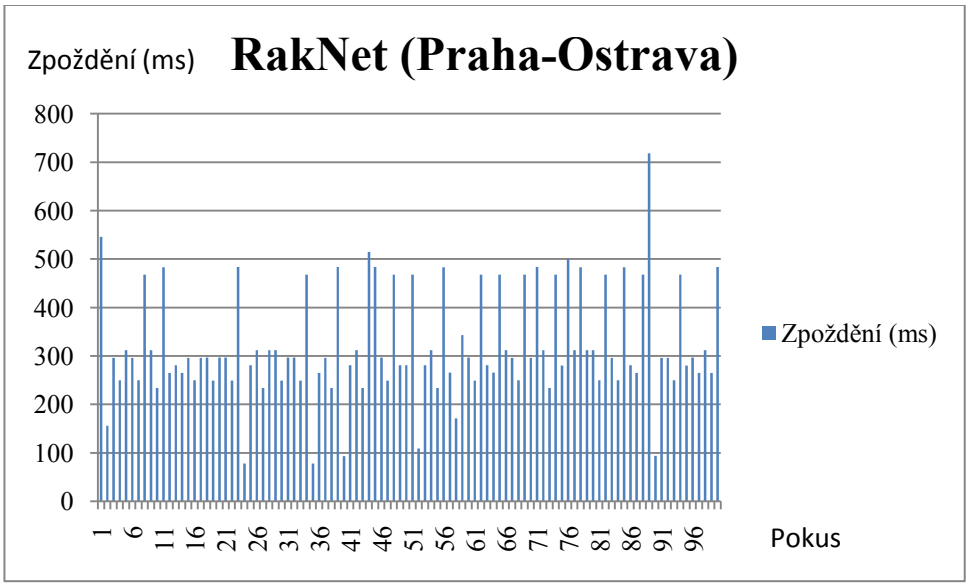
Knihovna DirectPlay v těchto testech příjemně překvapila. Obvyklá doba odezvy se pohybovala mezi 60-80ms. Až na občasné výkyvy se tak jedná o velmi dobré výsledky. Kromě těchto výkyvů totiž nepřesáhla doba zpoždění hodnotu 100ms.



Graf XI: DirectPlay (Praha-Ostrava)

4.2.3 RakNet

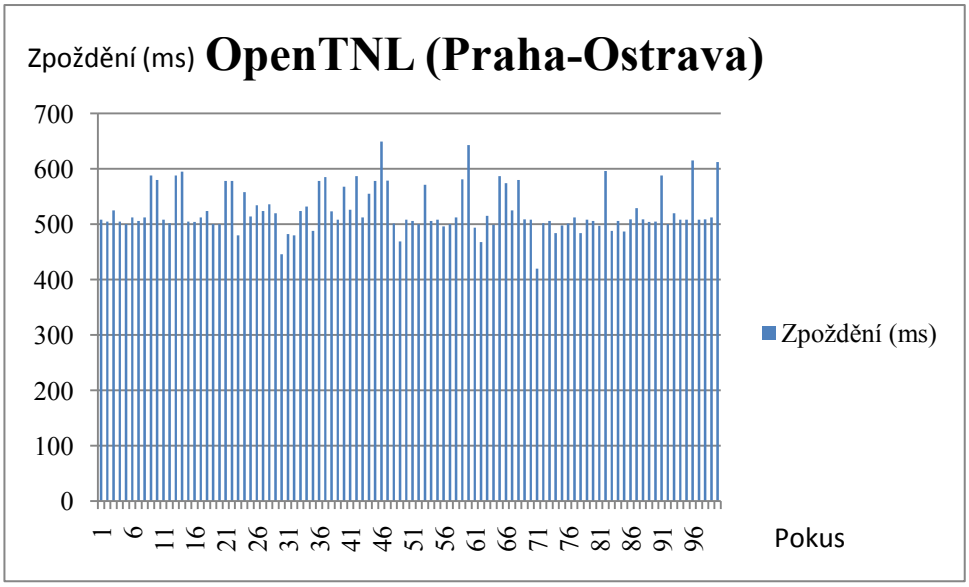
Knihovna RakNet se ukázala při testech mezi Prahou a Ostravou jako nestabilní. Z následujícího grafu jsou patrné značné výkyvy a nestabilita. V průměru se doba odezvy pohybovala v rozmezí mezi 250 až 300ms. Byli však poměrně časté výkyvy, dosahující hodnot až 500ms, nebo naopak klesající k hodnotám okolo 100ms.



Graf XII: RakNet (Praha-Ostrava)

4.2.4 OpenTNL

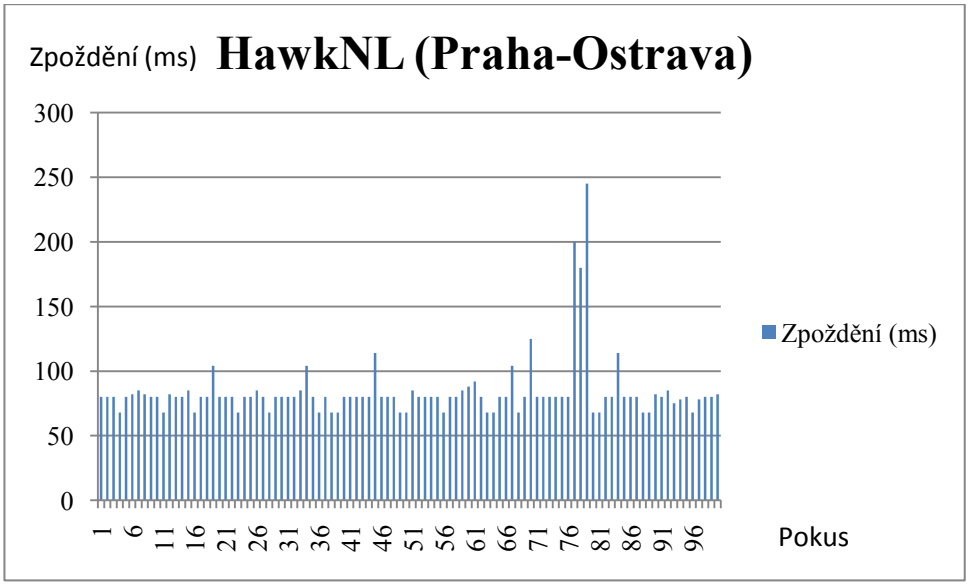
OpenTNL dosáhla na rozdíl od knihovny RakNet o něco stabilnějších výsledků, které se však pohybovali v průměru okolo 500ms. Takovéto zpoždění je již velmi citelné. Vzhledem ke špatným výsledkům při testech na lokálním počítači se však nejedná o žádné překvapení.



Graf XIII: OpenTNL (Praha-Ostrava)

4.2.5 HawkNL

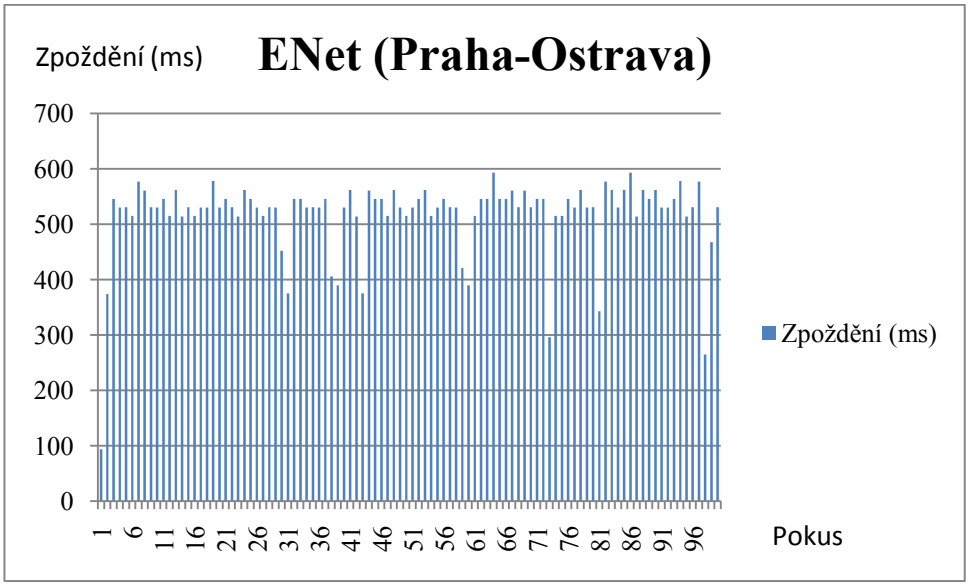
Knihovna HawkNL dosáhla v těchto testech velmi dobrých výsledků. Průměrná doby odezvy se pohybovala okolo 80ms, a jedná se tak o velmi dobrý výsledek. Až na několik výkyvů se jednalo o velmi stabilní časy.



Graf XIV: HawkNL (Praha-Ostrava)

4.2.6 ENet

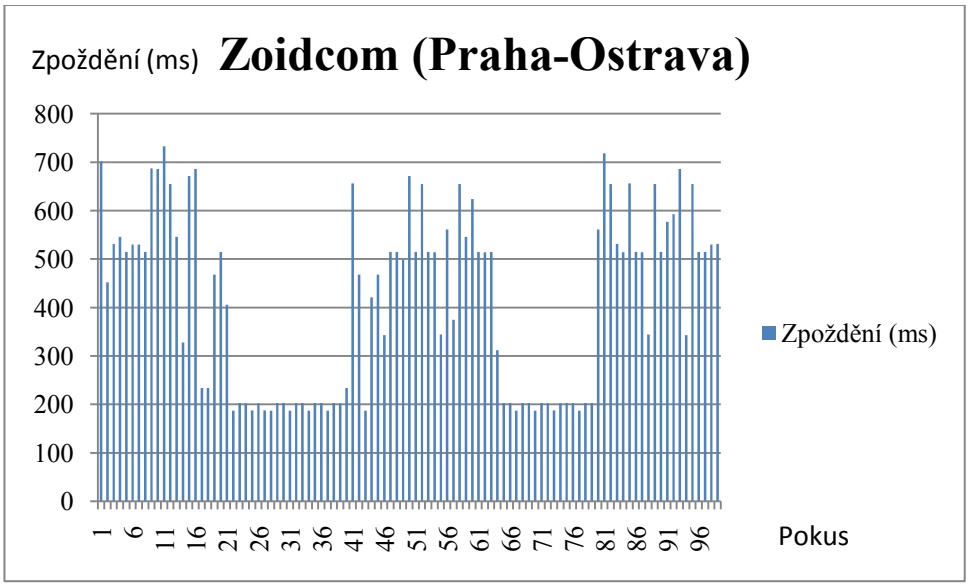
Ačkoliv dosáhla knihovna ENet v testech na lokálním počítači velmi dobrých výsledků, u testů na větší vzdálenost neuspěla. Ukázala se jako poměrně nestabilní. Doba odezvy se pohybovala ve většině případů okolo 500ms, v některých případech pak klesala ke 300ms. Jedná se zřejmě o důsledek kontroly doručených packetů, který se při testech na lokálním počítači neprojevil.



Graf XV: ENet (Praha-Ostrava)

4.2.7 Zoidcom

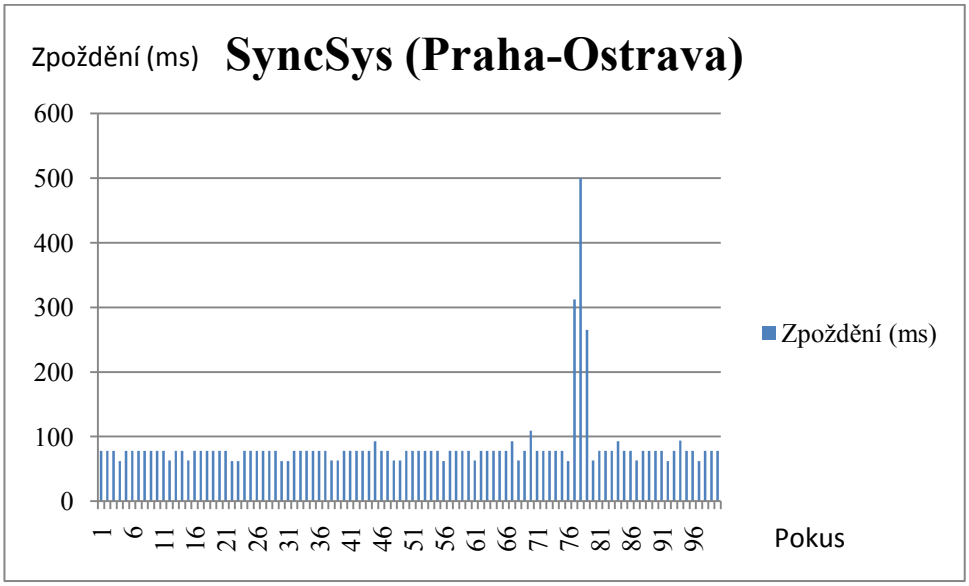
Knihovna Zoidcom dosáhla také velmi zajímavých výsledků. Místy se rychlost odezvy pohybovala stabilně okolo 200ms, jindy však vyskočila až nad 500ms. Takovéto výkyvy se vyskytovaly v pravidelných intervalech.



Graf XVI: Zoidcom (Praha-Ostrava)

4.2.8 SyncSys

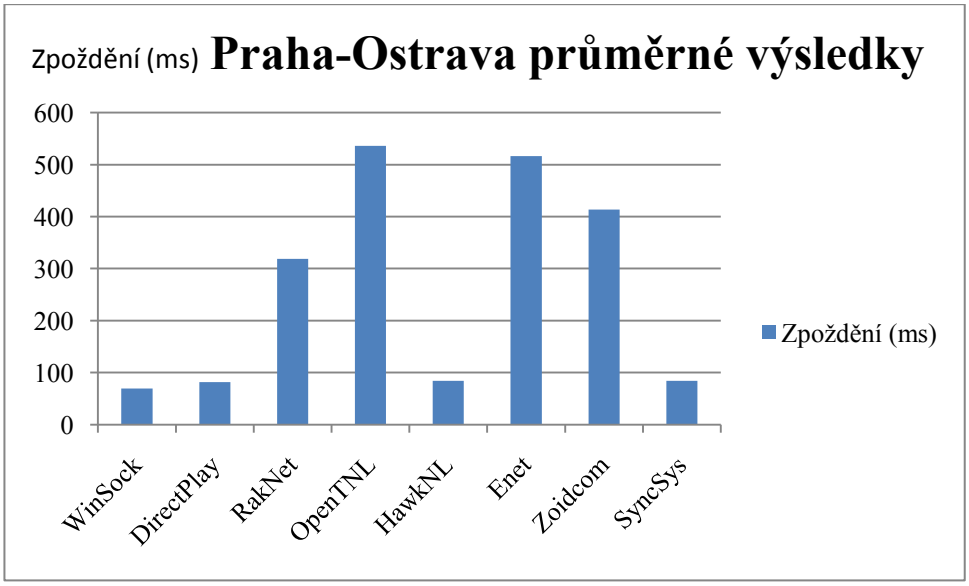
Poslední z testovaných knihoven SyncSys dosáhla velmi dobrých výsledků. Doba odezvy se pohybovala stabilně na 78ms. Až na několik výkyvů se jednalo o nejstabilnější výsledky, blížící se ping hodnotě. Obhájila tak kvalitní výsledky na lokálním počítači a dokázala, že je jednou z nejrychlejších knihoven.



Graf XVII: SyncSys (Praha-Ostrava)

4.2.9 Průměrné výsledky

Následující graf obsahuje průměrné výsledky z jednotlivých testů. Jedná se o celkový přehled testů na virtuální síti mezi dvěma počítači umístěných v Praze a Ostravě.



Graf XVIII: Praha-Ostrava průměrné výsledky

Výsledky se až na několik výjimek podobají výsledkům z testů na lokálním počítači. Největšího rozdílu dosáhla knihovna ENet, která se velmi pohoršila, a v zátěžových testech mezi Ostravou a Prahou dosáhla jednoho z nejhorších výsledků. Naopak si polepšila knihovna DirectPlay, která se může zařadit v testech na větší vzdálenost k těm nejlepším.

4.3 Shrnutí testů

Kdo by tedy logicky očekával, že odezva všech síťových knihoven bude na lokálním počítači předpokládaných 0ms, mohl by být těmito výsledky překvapen. U některých knihoven dosahují odezvy až zarážející velikost. U některých knihoven se jedná pravděpodobně o důsledek kontroly odesílaných packetů UDP.

Za zmínku také stojí, že testy mohou být subjektivní. Není totiž 100% zaručeno, že při implementaci testů nenastala chyba. Se všemi knihovnami lze pracovat více způsoby. Některé mohou případně upravit její výkon. Všechny testy však byly zpracovány velmi svědomitě podle přiložené dokumentace (případně podle ukázkových příkladů). Je však možné, že různé optimalizace by mohly vést k různým výsledkům. Nepředpokládá se však, že by se rapidně lišily od následujících závěrů.

Nejlepších výsledků dosáhla základní knihovna WinSock, následovala ji knihovna SyncSys a HawkNL. Jejich časy se blížily vždy hodnotě ping. Také DirectPlay se ukázala jako schopná při testech jak na jednom počítači, tak na větší vzdálenosti.

Zajímavým překvapením se stala knihovna ENet, která dosahovala na lokálním počítači velmi dobrých výsledků, přesto při testech na vzdálený počítač pohořela.

Ostatní knihovny v testech neuspěly. Knihovna RakNet dosahovala na lokálním počítači velikost odezvy až 100ms, a při testech na virtuální síti se ukázala jako nestabilní. OpenTNL dosáhla velmi špatných výsledků v obou testech. Podobně na tom byla i knihovna Zoidcom.

4.4 Porovnání a závěr

Jak je z výsledků testů patrné, mnohé z knihoven dosahují při síťové komunikace velmi vysokou latenci. Otázkou je, zda kontrola UPD packetů a další mechanismy dané knihovny, vyrovnají negativa těchto zpoždění.

V případě, že by se jednalo o souborový systém, který by se staral o odesílání souborů a kontrolu, zda všechny části byly v pořádku odeslány, bylo by vzniklé zpoždění malou daní za ověřený přenos dat. U počítačových her však jde především o rychlost, s jakou herní klient komunikuje se serverem. Otázkou tak zůstává, zda klient chce mít ověřeno, že data byla v pořádku doručena. V případě počítačových her lze však počítat s možnou ztrátou dat při přenosu a přizpůsobit se tomuto problému.

Nejlepších výsledků nakonec dosáhla základní knihovna WinSock. Ačkoliv poskytuje základní síťové operace, je také velmi flexibilní a snadno použitelná. Jako další vhodná knihovna se jeví již pouze SyncSys, která dosáhla velmi dobrých výsledků v testech a je dostupná zdarma.

Knihovna DirectPlay také nedosahovala špatných výsledků, přesto je však již považována za zastaralou, a tudíž ji nelze doporučit. Poslední knihovna, která měla velmi dobré výsledky je HawkNL. Lze ji však považovat za nevhodnou vzhledem k chybějící podpoře a několikátému stáří.

V následující tabulce budou stručně shrnuty výhody a nevýhody jednotlivých knihoven po testech. Jedná se o rychlý přehled, který může usnadnit správnou volbu knihovny.

Název	Výhody	Nevýhody
DirectPlay	Rychlá odezva	Zastaralé (DirectX 8.1) Omezení pouze na OS Windows
RakNet	Kvalitní zpracování a dokumentace Podpora od vydavatele	Velká latence Poplatek v komerčních aplikacích
OpenTNL	Zdarma i pro komerční využití	Velká latence Nekvalitní dokumentace
HawkNL	Rychlá odezva Zdarma i pro komerční využití	Zastaralé Nekvalitní dokumentace
ENet	Zdarma i pro komerční využití	Velké latence Pouze základní funkce
Zoidcom	Kvalitní zpracování a dokumentace	Velká latence Poplatek v komerčních aplikacích
SyncSys	Rychlá odezva Zdarma i pro komerční využití	Chybějící tutoriál

Tabulka III: Shrnutí síťových knihoven

4.4.1 DirectPlay

Ačkoliv je již knihovna DirectPlay zastaralá a v dnešní době se již nepoužívá, obstála v testech poměrně dobře. Je jen otázkou, co vedlo vývojáře z Microsoftu z vyloučené síťové podpory z DirectX. Pravdou však zůstává, že samotná implementace působí lehce nepřehledně a dokumentace není také optimálně zpracována (k dnešnímu dni není možné najít kvalitně zpracovanou dokumentaci případně tutoriály, vyjma těch, které jsou přiloženy jako součást DirectX SDK).

4.4.2 RakNet

Jedná se o velmi kvalitně zpracovanou a zdokumentovanou knihovnu, která umožní okamžitý start do síťové komunikace i úplným začátečníkům. U zátěžových testů však neobstála. Může se však o vhodnou volbu v případě, že není potřeba rychlá odezva, ale spíše ověřený přenos dat. RakNet také poskytuje nadstandardní funkce, oproti většině ostatním testovaným knihovnám. V komerčních aplikacích je však nutné počítat s poplatkem.

4.4.3 OpenTNL

OpenTNL u zátěžových testů neobstála. Také vzhledem k její nekvalitní dokumentaci a webové prezentaci je lepší se porozhlédnout po kvalitněji zpracované knihovně, kterých existuje v tomto oboru celá řada.

4.4.4 HawkNL

Ani tato knihovna by nebyla nejlepší volbou. Ačkoliv dosáhla v zátěžových testech dobrých výsledků, je již velmi zastaralá a neexistuje k ní jakákoliv podpora. Pro vývoj her však není nepoužitelná, ale rozhodně se dají najít lepší varianty.

4.4.5 ENet

Tato knihovna neobstála u zátěžových testů. Její odezva na lokálním počítači byla minimální, ale u testů na vzdálený počítač neuspěla. Obsahuje kvalitní a přehlednou dokumentaci a pracuje se s ní příjemně. Neposkytuje však žádné speciální výhody, kvůli kterým by převyšovala ostatní knihovny. Pro základní síťovou komunikaci však bohatě postačuje. Díky velkým latencím však není vhodná do počítačových her.

4.4.6 Zoidcom

Ačkoliv Zoidcom v zátěžových testech neobstála, jedná se o poměrně kvalitní knihovnu, kterou je možné využít při základní síťové komunikaci. Do počítačových her se však díky své vysoké odezvě nehodí.

4.4.7 SyncSys

Tato knihovna je vytvářena jako vhodná pro vývoj MMO her, a proto není divu, že u zátěžových testů obstála. Její odezva je více než dobrá, a její neomezené použití u komerčních aplikací z ní dělá vhodnou volbu. Její jedinou nevýhodou může být nepříliš kvalitní dokumentace.

5 Popis síťových her

Tato kapitola se bude věnovat základní teorii, věnující se problémům při vývoji síťové komunikace u počítačových her. Budou zde shrnuty základní problémy a jejich možná řešení. Nebude zde rozebírán každý problém do podrobností, spíše se nastíní základní řešení daného problému. Nejsou zde rozepsány všechny způsoby, jak lze vytvořit multiplayer ve hře, jsou zde však popsány některé z možností, jak lze k tvorbě těchto her přistupovat.

Specifickou vlastností her pro více hráčů je umístění autority (řídící logiky hry). Tím je myšlena ta část hry, která určuje pravidla hry, vstupy od jednotlivých klientů, a rozhoduje o správné synchronizaci všech připojených klientů. Tímto problémem se podrobněji zabývá kapitola 5.1.

Základním problémem je synchronizace mezi klientem a serverem. Při tvorbě síťových her je potřeba u klienta udržovat co nejaktuálnější pozice všech ostatních hráčů a objektů. Tato synchronizace je základním kamenem síťové hry. Jak komunikuje klient se serverem je rozebráno v kapitole 5.2.

Dalším z problémů je latence mezi klientem a serverem. Při přenosu dat dochází ke zpoždění, se kterým je nutné počítat. Vzhledem k tomu, že optimalizovat přenosovou rychlost mezi dvěma síťovými prvky většinou není ani možné, jsou nutná řešení na straně klienta (nebo i serveru), která se těmito prodlevami vypořádají. Jak se vypořádat se zpožděním vzniklým síťovou komunikací je vysvětleno v kapitole 5.3.

5.1 Určení autority

Autoritou je myšlen vybraný počítač (na kterém je spuštěná konkrétní hra), který rozhoduje o událostech vzniklých ve hře. Zde jsou obsloužena pravidla hry, data od klientů a další. Na tomto počítači je tak rozhodnuto, kdy například střela zasáhne hrajícího hráče a odešle na ostatní klienty tuto informaci. Je zde také udržovaná aktuální definice světa, která má absolutní přednost před definicí světa na ostatních zařízeních (např. pozice hráčů). Ošetřují se tak možné podvody, kterých by se mohli hráči na ostatních zařízeních dopustit (tzv. cheaty).

Pokud se jedná o hru, primárně tvořenou hru více hráčů, je dobré zvolit autoritu na serveru. V případě, že není možné z nějakých důvodů vytvořit server, který bude obsahovat logiku síťové hry, je nutné zvolit některého z klientů za autoritu (nejlépe klient s nejmenší latencí na server). Tímto se klient stává autoritou a rozhoduje o všech rozhodujících momentech hry.

Klient ale vždy komunikuje výhradně se serverem (tedy standardní komunikace client - server). Komunikace klientů mezi sebou (peer-to-peer) není vhodná, protože server obsahuje vždy alespoň základní logiku i v případě, že autoritou je jiný klient. V tomto případě server sám určí, kde budou zpracována pravidla hry, a stará se o správné rozesílání packetů.

5.1.1 Autorita na straně serveru

V případě, že je autorita na serveru, odstraní se tak mnoho problémů oproti autoritě na straně klienta. Daný server vytvoří novou instanci světa a čeká na vstupy od klientů. Každému klientovi, který se následně připojí do hry, se odešle kompletní (a aktuální) definice světa ze serveru. Klient následně odesílá vstupy (z klávesnice, myši, mikrofonu, atd.) a přijímá informace ze serveru. Veškeré akce

klienta jsou tak vykonány na straně serveru, a u klienta je pouze zobrazen výsledek na výstupní zařízení (např. animace, střely, …). Tím však vzniká zpoždění mezi vstupem na klientovi a následném zobrazení. Tímto problémem se zabývají kapitoly 5.2 a 5.3.

Pokud je takto zvolena logika hry na serveru, odstraní se tím řada problémů se synchronizací klientů, neboť existuje pouze jedna dominantní verze herního světa. Nevýhodou však je nutnost ošetření vzniklých latencí u klienta (mezi událostí na vstupu a zobrazením u daného klienta).

Tento princip je vhodný zejména u her, kde se veškeré akce odehrávají v reálném čase, a je nutná úplná synchronizace mezi všemi klienty.

5.1.2 Autorita na straně klienta

V případě, že není možné vytvořit server jako autoritu, je nutné určit na hlavní pozici jednoho z klientů (nejlépe klient s nejlepším připojením na server). Server se pak stává pouze zprostředkovatel síťové komunikace mezi všemi klienty. To však přináší řadu nevýhod.

Logika hry vždy běží na straně klienta, a ten pouze odesílá informaci o pozici v daném světě. Může se tak stávat, že světy u jednotlivých klientů nejsou synchronizovány. Musí se tak kontrolovat konzistentnost jednotlivých definic světů mezi všemi hráči. To přináší další zátěž na síťové rozhraní.

Klient, kterému je přidělena autorita, se musí postarat o samotnou logiku hry. Pracuje na základně pravidel hry a rozhoduje na základě informací od všech klientů (např. rozhoduje, který hráč dřív sebral nějaký předmět, detekci hráče se střelou apod.). Výsledky svých rozhodnutí pak musí předat všem klientům. Musí se tak nejen synchronizovat všichni klienti mezi sebou, ale také vůči klientovi s autoritou (případně lze synchronizovat všechny klienty výhradně přes tohoto klienta – neprobíhá tak synchronizace mezi klienty navzájem, ale určený klient zastává roly serveru). V první řadě se tak zvýší latence, neboť informace na toho klienta musí jít vždy přes server, a síťová komunikace mezi ním a serverem je tak poměrně zatížena (každá informace od ostatních klientů musí být odeslána tomuto klientovi).

Další nevýhodou je horší detekce podvodů, neboť v případě, že logika hry běží u každého klienta zvlášť, není zaručeno, že není pozměněn kód hry, který mění pravidla hry (např. hráč se pohybuje vyšší rychlostí apod.). V případě autority na serveru lze jednoduše zkontrolovat stav světa mezi klientem a serverem a odhalit tak nesrovnalosti.

Tento princip je tak vhodný pouze u jednoduchých her, u kterých se nepočítá se velkým množstvím hráčů, případně u her, kde se nemusí rozhodovat v reálném čase (např. tahové hry apod.).

5.2 Synchronizace hry

Klient komunikuje se serverem pomocí malých balíčků dat. Ty jsou odesílány s vysokou frekvencí, aby byla u klienta byla vždy nejaktuálnější verze světa, nebo aby se server se v co nejkratší době dozvěděl o akcích na straně klienta. Frekvence přenosu dat mezi klientem a server závisí na typu hry a kvalitě připojení. Například v případě serverů postavených na Source enginu se jedná standardně o 20 balíčků za sekundu ze serveru na klienty, a 33 balíčků za sekundu z klienta na server [8].

Síťová komunikace má omezenou kapacitu. Z těchto důvodů je vhodné odeslat kompletní definici světa na klienta pouze po prvním přihlášení (nebo v případě desynchronizace). Poté již stačí odesílat pouze změny, které se za určitý časový interval udály. Tyto změny jsou odesílány ze serveru v pravidelných interval na všechny klienty.

Při síťové komunikaci však vznikají prodlevy, ať už se jedná o komunikaci serveru s klientem nebo naopak, je vždy nutné počítat s nějakým zpožděním. Toto zpoždění je různé u jednotlivých klientů. Tyto rozdílné časy mohou způsobovat logické problémy. Proto je důležité, aby všechny připojené počítače byly synchronizovány na stejný čas. K tomuto účelu může posloužit časovač a společné hodiny, které jsou součástí dat odeslaných ze serveru na klienta. Pomocí sdíleného času a synchronizovaným časovačem lze dosáhnout konzistence i v případě, kdy se zpoždění u jednotlivých klientů liší.

V případě, že klient nemá kvalitní připojení a prodleva mezi serverem a klientem je vysoká a velký objem dat na server by zapříčinil ještě větší zpoždění, lze na daného klienta odesílat omezený počet aktualizací. Pomocí časovače zůstane zaručeno, že i když klient obdrží méně informací ze serveru (v menších frekvencích), zůstane synchronizovaný. Je tak zaručeno spravedlivé hraní i pro hráče s horším připojením.

5.2.1 Server

Hodiny a časovač slouží pro synchronizaci výpočtů na straně serveru, nebo pro vykreslování objektů na výstupním zařízení na straně klienta. Na serveru se v těchto intervalech vždy provede daný výpočet (například se zpracují vstupy od klientů, provede se pohyb hráče po světě, detekce kolize, apod.). Po každém výpočtu se provádí kontrola, jestli je potřeba odeslat aktualizovaný stav světa na klienty (případně se určí, na kterých klientech je potřeba provést aktualizaci). Vyšší frekvence časovače zajistí vyšší přesnost při výpočtech nad aktuálním světem, ale výrazně zvýší zátěž výpočetního výkonu počítače. Ideální interval závisí na samotné hře, obecně jak však vhodné nastavit časovač pro tyto výpočty okolo 60 výpočtů za sekundu.

5.2.2 Klient

Klient odesílá na server příkazy, který hráč během hraní provedl (převážně události ze vstupních zařízení). Aby se však neodesílala data na server po každé hráčově interakci (například pohyb myši, stiskl klávesy, atd.), odešle se balíček více těchto příkazů (v případě, že nastane více vstupů na straně klienta) na server (viz. úvod kapitoly 5.2). Tyto data se odesílají v určitý čas, podle synchronizovaných hodin. Tyto požadavky jsou na straně serveru zpracovány. V následující aktualizaci klient obdrží definici světa, která obsahuje i reakci na jeho vstupy. Zde však vzniká zpoždění, které by zapříčinilo opožděné vykreslení výsledku hráčových vstupů na monitor. To by vedlo k neplynulosti hry a hra by byla v podstatě nehratelná. Tento problém se dá řešit několika způsoby a jsou podrobněji popsány v kapitole 5.3.

5.3 Problémy latence

Jak již bylo zmíněno v kapitole 5.2.2, v případě, že autoritou je server a provádí veškeré výpočty spojené s hráčovým vstupem, nastává u klienta zpoždění, které vede k opožděnému vykreslování hry v závislosti na jeho vstupech. Dalším problémem je opoždění aktuálního stavu světa. Ke klientovi se

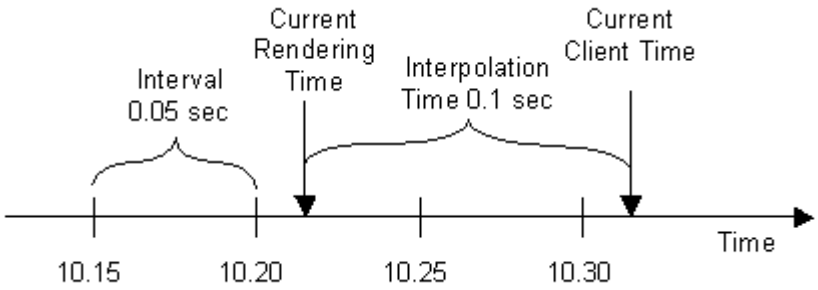
totiž dostane verze světa s jistým zpožděním, což může vést k dalším problémům (například opožděná reakce hráče na nějakou událost apod.).

Mezi techniky, které dané problémy řeší, patří interpolace, předpovídání vstupů a kompenzace zpoždění. Ty jsou popsány v následujících podkapitolách a stručně popisují možné způsoby řešení, jak se vyrovnat s tímto problémem.

5.3.1 Interpolace

Klient obdrží ze serveru aktuální svět v pravidelných intervalech (viz. kapitola 5.2). Ty však nepřichází v takové frekvenci, v jaké se vykresluje daný svět na výstupu (standardně 60 snímků za sekundu), ale vždy s jistým zpožděním. Dále může nastat ztráta dat při síťovém přenosu. Pokud by došlo ke ztrátě dat, a vykreslovalo by se vždy podle přijatých dat ze serveru, působil by obraz trhaně. Také by byl opožděný oproti stavu světa na serveru.

Z těchto důvodů se používá technika tzv. interpolace. Jedná se o způsob, jak vypočítat následující kroky pro vykreslení na základě předchozích kroků (pozice objektů, animace, …). V paměti se udržuje několik posledních stavů světa, podle nich se dá při každém vykreslení interpolovat z posledních dvou známých pozic, a získat tak předpokládanou pozici pro následující výstup na monitor.



Obrázek III: Interpolace [8]

Na předchozím obrázku je znázorněn časový průběh vykreslování na obrazovku. Interval znázorňuje časové úseky, ve kterých přichází data ze serveru (v tomto případě v intervalu po 50ms). Pokud by v čase 10.25 nepřišla aktuální data, pomocí interpolace by se vypočítaly (na základě stavu světa v čase 10.20 a 10.15) aktuální pozice objektů. Tím se docílí plynulosti animací.

V případě, že dojde ke ztrátě několika balíčků ze serveru a klient by už neměl potřebné informace pro interpolaci (např. by došlo k velké ztrátě dat ze serveru), musí se klient znovu kompletně synchronizovat se serverem (stáhnout opět kompletní definici světa).

Tato technika tak umožňuje plynulou hru na straně klienta i v případě, že data nejsou ze serveru odesílána v pravidelných a dostatečně rychlých intervalech, umožňující plynulé vykreslování (čili s frekvencí menší jak 30-60 za sekundu). Dále také odstraňuje problém se zpožděním, případně i při ztrátě dat ze serveru.

Tato technika je využívána především pro vykreslování pohybujících se hráčů a jiných objektů. Je tedy určena především pro animace.

5.3.2 Předpovídání vstupů

Jestliže je veškerá logika umístěná na serveru, vzniká zpoždění mezi hráčovým vstupem a vykreslením na jeho výstupním zařízení. Tento problém lez vyřešit tak, že klient provede vykreslení světa (například pohyb hráče, myši apod.) v závislosti na jeho vstupech (podle stejných pravidel, jako jsou na serveru). Klient tak předpovídá, jaké operace se provedou na serveru. Zde však může nastat desynchronizace s aktuálním světem na serveru.

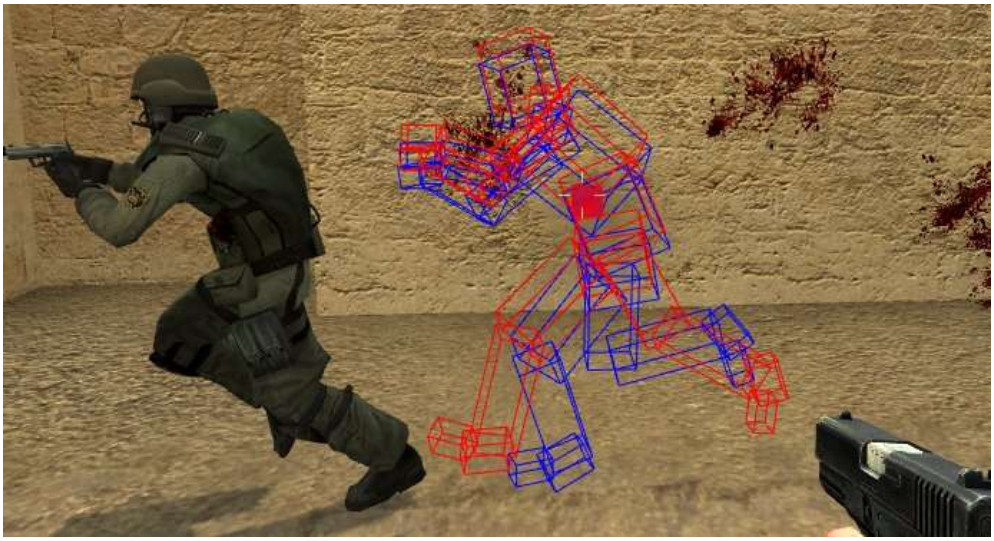
Klient tedy obdrží s jistým zpožděním aktuální verzi světa ze serveru (obsahující aktuální pozici hráče) a porovná ji s tou předpovídanou u klienta. Pokud je mezi těmito dvěma verzemi rozdíl, upraví se aktuální pozice hráče podle stavu na serveru (server je hlavní autorita a má vždy přednost).

Tato technika funguje pouze v případě, že klient zná stejná pravidla jako server. Dá se tak předpovídat pouze hráčův pohyb po světě a jeho interakce s objekty. Je však velmi důležitá pro plynulé hraní hry.

5.3.3 Kompenzace zpoždění

Poslední ze zmíněných technik je kompenzace zpoždění. K té dochází například v případě, kdy hráč vystřelí na jiného hráče, který je v pohybu. Výstřel jednoho hráče je odeslán z klienta na server, kde dorazí s jistým zpožděním. Mezitím se však již druhý hráč stačil pohnout na jinou pozici, a server by tuto situaci vyhodnotil jako špatný zásah, ačkoliv hráč na klientovi mířil přesně.

Na straně serveru tak musí dojít ke kompenzaci zpoždění. Server musí uchovávat informace o pozici hráče z několika posledních okamžiků (např. za poslední 1 sekundu). Požadavek od klienta, který s sebou nese také časovou informaci o vzniku události (pomocí synchronizovaných hodin, viz. kapitola 5.2) se na straně klienta zpracuje tak, že se dohledá, jaké byli pozice hráčů v daném čase. Zajistí se tak správná detekce střely s jiným hráčem podle stavu v čase výstřelu. Následující obrázek vykresluje názorný příklad kompenzace zpoždění (snímek pořízen ze hry Counter-Strike 1.6 [8]).



Obrázek IV: Kompenzace zpoždění [8]

Vykreslená postava znázorňuje aktuální pozici na serveru. Server však přijme vstup od klienta, ve kterém je časové razítko z doby, kdy klient viděl hráče na předchozí pozici (znázorněna jako obrys postavy červenou barvou). Server tak musí dopočítat, v jaké byl pozici hráč v daném čase výstřelu (znázorněno jako obrys postavy modrou barvou). Tyto dvě pozice se nemusí úplně shodovat (na obrázku je vidět rozdíl v pozicích mezi červeným a modrým obrysem), je však důležité, aby byl výsledek co nejpřesnější. Tím je zajištěno, že ačkoliv je stav světa na serveru o několik kroků napřed, dá se zajistit správný zásah i při zpoždění.

To může mít však za následek tzv. paradox [8]. Ten nastává v případě, kdy u hráče je zobrazeno, že je již v bezpečí (např. schovaný za překážkou, hráč již doběhnul za roh apod.), ale následně je vrácen o několik pozic zpět, kde je zasažen. Tento paradox však není možné nějakým obecným způsobem řešit, neboť mezi klientem a server je vždy nějaké zpoždění, které tento paradox zapříčiňuje.

5.4 Shrnutí a závěr

Jak již bylo uvedeno v úvodu této kapitoly, jedná se pouze o jedny z možných způsobů, jak přistupovat k tvorbě síťové hry. Způsob zvolení autority na straně serveru, který zpracovává veškeré vstupy od klienta, je vhodný zejména u her, které pracují v reálném čase a je zapotřebí co nejspravedlivějšího zpracování informací pro všechny klienty. U tohoto typu řešení zde byly rozebrány některé z jeho problémů a techniky, jak se s těmito problémy vypořádat.

Síťová komunikace však není nikdy úplně dokonalá, a tak je zapotřebí počítat s některými nepřesnostmi, které v těchto hrách mohou nastat (viz. paradox v kapitole 5.3.3). Pokud je však implementace síťového kódu kvalitní a tyto problémy minimalizovány, lze přistupovat k těmto hrám s určitou shovívavostí.

6 Implementace síťové hry

V této kapitole bude popsána implementace jedné, velmi jednoduché síťové hry, vytvářené v jazyce C++. Budou zde popsány jednotlivé kroky v pořadí, v jakém následovali při vývoji. Budou zde popsány výhody a nevýhody, případně chyby, které při vývoji vznikly, a návrhy na možné optimalizace.

Jedná se o velmi jednoduchou 2D plošinovou hru pro více hráčů. Každý hráč může střílet barevné koule (střely), které po zasažení ubírají ostatním hráčům životy. Pokud hráčův život klesne na 0, umírá, a objevuje se opět na náhodném místě. Hra je omezena na malou arénu, ve které se veškeré souboje odehrávají. Hráči mohou skákat na různé plošinky a pohybovat se libovolně po tomto vymezeném prostoru.

Tvorba hry probíhala postupně (pomocí iterací). Od počátku se již počítalo s tím, že se bude jednat o síťovou hru pro více hráčů, tudíž se mohl vývoj tomuto zadání přizpůsobit. Jak se však později ukázalo, některé části vývoje nebyly zcela optimální. Vysvětlení a návrhy na optimálnější řešení budou popsány v závěru této kapitoly.

Každé iteraci je věnována samostatná kapitola, které se danému kroky věnuje. Nejprve byla vytvořena grafická stránka hry, tedy především animace postav. V tomto případě se jednalo o základní pohyb postavy ve 2D. Dalším krokem pak bylo vytvoření definice světa (jednoduché překážky) a detekce kolize mezi hráčem, střelou a okolním světem. Následně se hra rozdělila na část klient/server, a každé se přidělila logika hry. V poslední části vývoje se implementovala síťová komunikace mezi klientem a serverem.

Účelem této práce nebylo vytvořit hru s dobrou hratelností, ale vytvořit základní prvky hry, které by simulovali hru pro více hráčů. Na těchto základech se však dá pokračovat a vytvořit tak hru s kvalitnějšími herními prvky. Pro samotnou ukázku síťové komunikace však takováto hra bohatě postačí.

6.1 Grafické rozhraní a animace

Veškerá grafická stránky byla vytvářena pomocí knihovny OpenGL a je vykreslena v pomocném nástroji GLUT. Veškeré scény jsou zobrazeny ve 2D projekční rovině (*gluOrtho2D*). Animace je rozpohybována pomocí tzv. skeletální animace (z anglického „skeletal animation“). Po vytvoření animací bylo přidáno vykreslení zbraně se zaměřovačem, pohyb střel a samotná definice světa.

6.1.1 Animace postavy

Jak již bylo řečeno výše, animace byla vytvořena pomocí tzv. skeletální animace. Celá postava je definována jako struktura kostí (viz. zdrojový kód I), které jsou navzájem provázány pomocí stromové struktury.

Struktura je definována pomocí těla (hlavní rodičovská kost), ze kterých vychází stehna a paže. Zatímco paže vychází z bodu na počátku těla, stehna vychází z bodu, určeného délkou kosti (spodní část těla). Pomocí délky tak lze vytvářet relativní pozice ostatních kostí. Každá paže a stehno má ještě svého vlastního potomka (ruce a nohy).

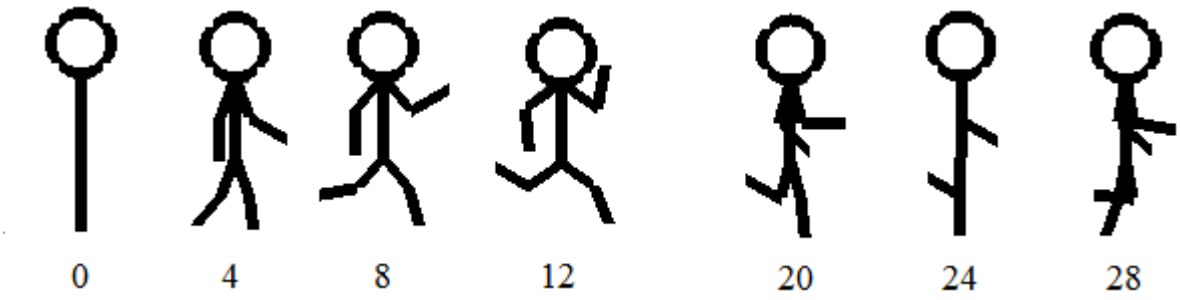
```
struct Bone {
    char name[20];
    float x, y; // pocatecni souradnice
    float uhel; // uhel proti ose x
    float delka; // delka kosti
    float uhelPosun;


    Bone* parent;

    int childCount;
    Bone* childrens[MAX_CHILDREN_COUNT];
};
```

Zdrojový kód I: Struktura kosti

Hlavní rodičovská kost definuje základní pozici hráče (jedná se o bod na vrcholu těla). Na rodičovskou kost jsou vázáni její potomci, kteří jsou závislý na její pozici a rotaci. Animace pak rekurzivně vykresluje všechny kosti s jejími potomky (důležité je zachování rotace předka). Pomocí této techniky lze dosáhnou plynulé animace postavy. Jediná část postavy, která nemá strukturu kosti, je hlava, která je však vykreslována v závislosti na pozici vrcholu hlavní kosti.

Pomocná proměnná *uhelPosun* slouží pro nastavení rychlosti animace, která probíhá v určitých krocích. S každým krokem se přičítá úhel dané kosti. Délka kosti se nikdy nemění. Pozice *x* a *y* jsou vždy odvozeny od předka dané kosti. Hlavní kost, která žádného předka nemá, určuje pozici samotného hráče ve světě. Při animaci se tak mění pouze úhly daných kostí a pro fyzický pohyb postavy se mění pouze pozice bodu *x* a *y* na základní kosti.

Na následujícím obrázku jsou vykresleny pozice hráče v různých fázích pohybu.



Obrázek V: Animace postavy

Každá fáze pohybu je rozdělena na 12 kroků (rozběh, doběh, …). Hráč se ze základního postavení (pozice 0) postupně rozbíhá. Pozice 12 znázorňuje finální postoj v běhu, po kterém se v podobných pozicích animuje zpět do základního postoje. Zde se však animace nedostává do bodu 0, ale zadní noha zůstává ve skrčené poloze. Animace pak vypadá mnohem plynuleji a vytváří se iluze reálnějšího pohybu postavy (viz. pozice 20, 24 a 28).

Výhodou skeletální animace je možnost přiřadit na jednotlivé pozice kostí textury, které upraví výsledný grafický efekt do příjemného uživatelského rozhraní. Tím je možné v budoucnu upravit hru, aniž by se nějakým zásadním způsobem muselo zasahovat do existujících zdrojových kódů. Další výhodou je možnost vytvořit detekci kolize na jednotlivé kosti, vhodné například pro určení

velikost zranění při zásahu určitého typu kosti (např. zásah do nohy ubere hráči míň života než do těla apod.).

6.1.2 Vykreslení zbraně a zaměřovače

Součástí postavy je také zbraň. Ta je pro zjednodušení zobrazena jako kanón na rameni hráče. Zobrazuje se také zaměřovač, tudíž je vidět, kam přesně hráč míří a jaká bude trajektorie střely. Zbraň se otáčí podle polohy myši na projekční rovině (vždy tedy míří na kurzor myši).

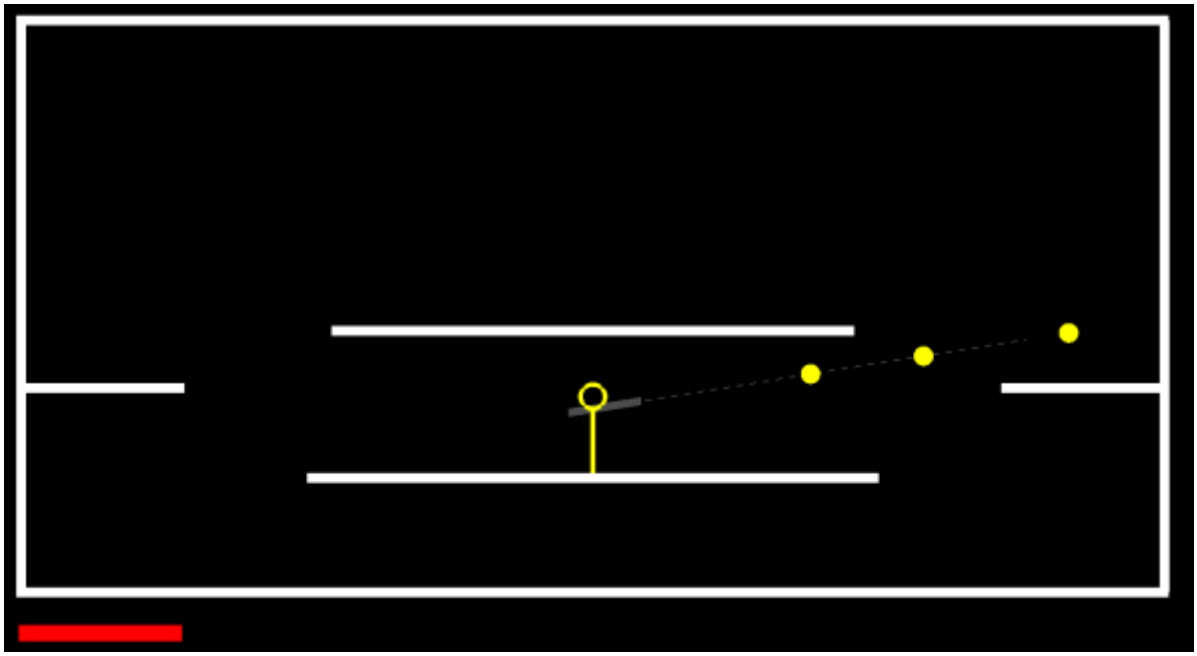
6.1.3 Střely

Střela je definována jako struktura, která obsahuje informace o počáteční, cílové a aktuální pozici. Cílová pozice se používá pro výpočet trajektorie, po které se daná střela pohybuje určitou rychlostí. Střely jsou vykreslovány jako barevné koule (podle barvy hráče). Při kolizi s nějakým objektem (např. objekt světa, hráč, apod.) střela mizí.

6.1.4 Definice světa

Definice světa je vykreslena pomocí velmi jednoduchých plošin. Ohraničují tak samotnou arénu, která vymezuje herní prostor. Jsou zde také plošiny, na kterých se dá pohybovat. Jedná se o strukturu, která pro jednoduché hraní vystačí.

Na následujícím obrázku je zobrazena definice světa (bílé plošiny) s jedním hráčem.



Obrázek VI: Ukázka ze hry

Je zde také vidět dříve popisovanou zbraň se zaměřovačem a střely. V levém dolním rohu je status života daného hráče.

6.1.5 Shrnutí

Grafická stránka hry je velmi jednoduchá. Pro jednoduchou hru, která slouží především pro testování síťové komunikace, však vystačí. Výhodou těchto struktur je možnost rozšíření o lepší grafické efekty, aniž by se musel měnit samotný princip animace (např. přidání textur apod.). Takto definované struktury také umožňují základní implementaci detekcí kolize apod.

6.2 Herní mechaniky a pravidla

Po vytvoření animace a grafického rozhraní bylo zapotřebí vytvořit ovládání, funkční pravidla hry a základního fyzikálního modelu. Jedná se především o detekci kolize mezi jednotlivými entitami ve hře (hráč, střela a objekt světa). Dále byla vytvořená simulace pádu hráče.

6.2.1 Ovládání

Ovládání je pevně dané (v budoucnu je možné ho upravit jako nastavitelné). Hra se ovládá klasicky pomocí myši a klávesnice. Pomocí myši hráč míří a střílí (stisknutím levého tlačítka myši se vystřelí). Pohybuje se pomocí tlačítek ‘A’ a ‘D’ doleva nebo doprava. Pomocí mezerníku se skáče.

Vstupy z kláves se zpracovává pomocí knihovny GLUT na straně klienta. Jedná se o jednoduché události, registrované na právě spuštěné herní okno.

6.2.2 Detekce kolize

Detekce kolize je provedena pomocí obdélníků, které obklopují jednotlivé objekty. Dále je rozdělena na dva základní typy: detekce kolize postavy s okolním světem a detekce střel.

Kolize všech objektů se iterativně kontroluje při každé změně definice světa. Tato iterace probíhá každých 30ms, při které se vypočítají nové pozice hráče a střel. Tento interval vede k plynulé animaci postav a jejich pohybu. Po každé změně pozice jsou následně provedeny testy na detekci kolize.

6.2.2.1 Kolize postavy s okolním světem

Jedná se o základní detekci kolize, která slouží především k samotnému pohybu hráče po světě a umožňuje základní pohyb. V případě, že není detekována kolize ve spodní části postavy (hráč se nedotýká nohami nějakého objektu), je ve stavu pádu. Při pádu se lze pohybovat pouze do stran (lze tedy směrovat pád). Jedinou výjimkou tvoří výskok, který se při dosažená horní hranice výskoku přemění na pád. Stojí-li tedy hráč pevně nohami na zemi, může se libovolně pohybovat a skákat.

Pokud hráč koliduje s některým z objektů světa ze stran, jeho pohyb se ukončí (jak při výskoku, tak při běhu). Při výskoku je také kontrolována pozice hlavy (při kolizi hlavy s plošinou se výskok okamžitě přepne do stavu pádu).

Tímto je docílena, poměrně jednoduchým způsobem, základní herní mechanika hry, která umožňuje pohyb hráče v dané definici světa.

6.2.2.2 Detekce střel

Druhým typem je detekce střel vůči ostatním objektům. Tím jsou myšleni hráči a objekty světa. Jakmile střela narazí do nějakého objektu, zmizí. V případě, že narazí do hráče, hráč přichází o malé procento zdraví (konkrétně o 10%).

6.2.3 Shrnutí

V případě ovládání již není potřeba žádných změn (maximálně možnost přidat volbu pro nastavení kláves pro pohyb a výskoky). Při detekci kolize však dochází ještě k drobným chybám. Ty však neomezují samotné hraní hry, a může být dodatečné upraveno v budoucnosti.

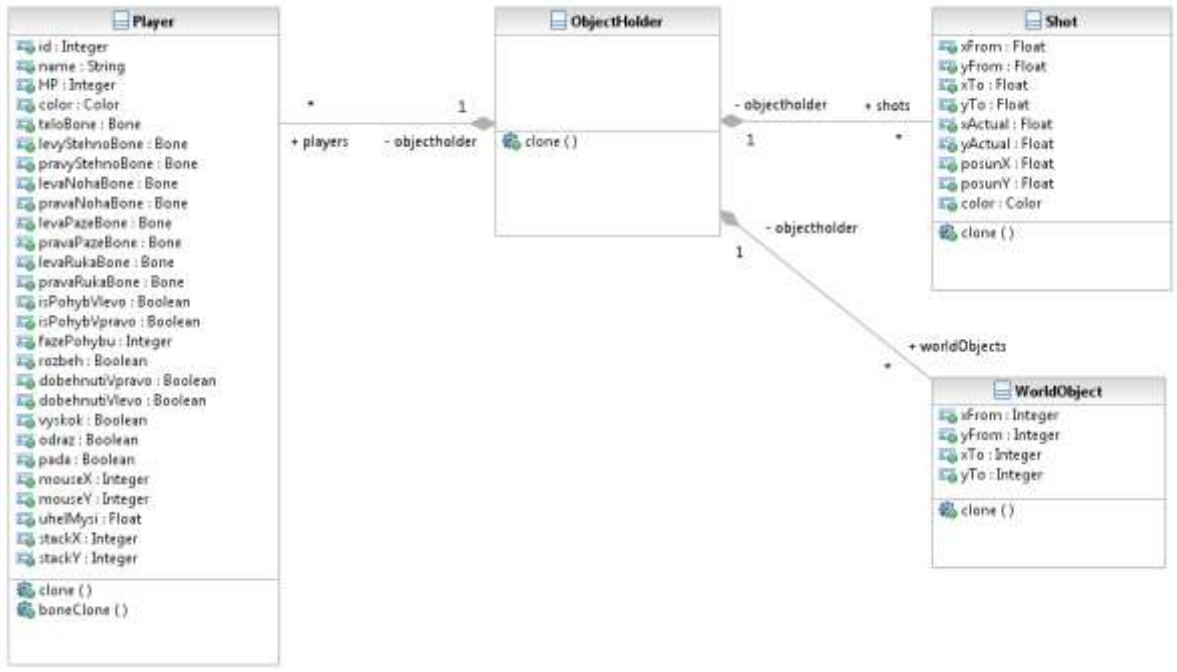
6.3 Rozdělení rolí

Po vytvoření grafické stránky hry a jejích mechanik se přistoupilo k samotné přípravě pro hru více hráčů. Bylo tak důležité rozhodnout, které operace budou prováděny na serveru a které na straně klienta. Byl zvolen přístup, kde autoritou se stává server, a klient slouží pouze pro ovládání a zobrazování hry (viz. kapitola 5.1.1).

Při implementaci se nejprve vytvořila simulace rozdělení na stranu klient/server. Prozatím se však nekomunikovalo po síti. Síťová komunikace byla přidána v pozdější fázi vývoje (viz. kapitola 6.4). Byla však učiněna důležitá příprava, která vedla ke správnému rozdělení rolí.

6.3.1 Úložiště dat

Nejprve je však nutné zmínit přístup k datům. Vzhledem k tomu, že server i klient pracuje se stejnými daty, byla vytvořena jednotná struktura, která obsahuje všechna data, týkající se herního světa. Následující třídní diagram zobrazuje tuto datovou strukturu.

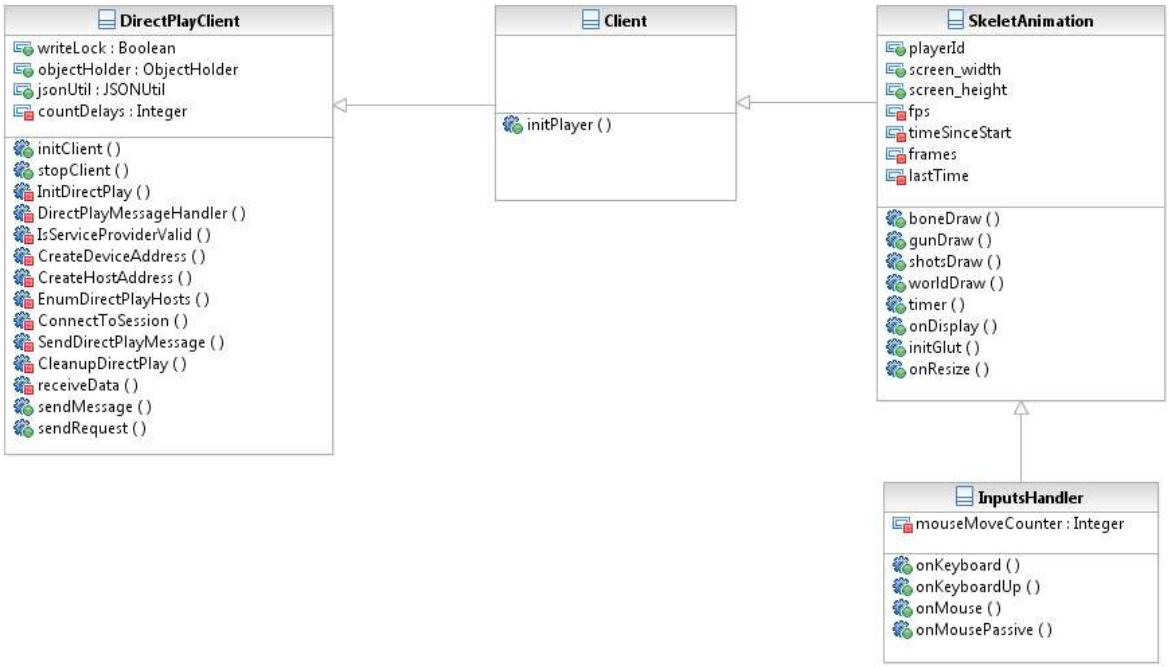


Obrázek VII: Třídní diagram – ObjectHolder

Součástí definice světa jsou tři datové struktury. Jedná se o hráče (*Player*), střely (*Shot*) a objekty světa (*WorldObject*). Všechny jsou vázány v jedné třídě *ObjectHolder*, která obsahuje kolekce jejich instancí. S touto strukturou se pracuje jednotně jak na straně klienta, tak na straně serveru. Je tak usnadněna práce s těmito objekty, neboť se ke všem datům přistupuje přes jednu třídu.

6.3.2 Klient

Jak již bylo zmíněno výše, na straně klienta se zpracovávají vstupy z klávesnice a myši a provádí se zobrazování aktuálního stavu světa na monitor. Pomocí knihovny GLUT je vytvořeno vizuální okno, ke kterému je též vázán posluchač na vstupní zařízení. Následující třídní diagram vykresluje hlavní třídy, které jsou určeny pro práci s klientem.



Obrázek VIII: Třídní diagram – Klient

Vše je zastřešeno třídou *DirectPlayClient*. Důležité je podotknout, že v této fázi vývoje ještě nebyla třída implementována v této podobě. Je zde však již zobrazen finální třídní diagram, ve kterém je již zahrnuta síťová komunikace. Při simulaci rozdělení na samostatné strany klienta a serveru však tato třída sloužila pro odesílání a příjem dat ze simulovaného serveru.

Dalším prvkem v hierarchii je třída *Client*, která dědí ze třídy *DirectPlayClient*. Ta obsahuje jedinou metodu *initPlayer*, která slouží pro obsluhu vytváření nového hráče. V této metodě se zpracovávají informace o hráči, které uživatel zadává při přihlášení do hry (jedná se o jméno a barvu). Dále se také vygeneruje náhodné *id* hráče, pomocí kterého je identifikován na serveru i na svém počítači. Pomocí rodičovské třídy se odesílá požadavek na server, kde se vytváří instance nového hráče.

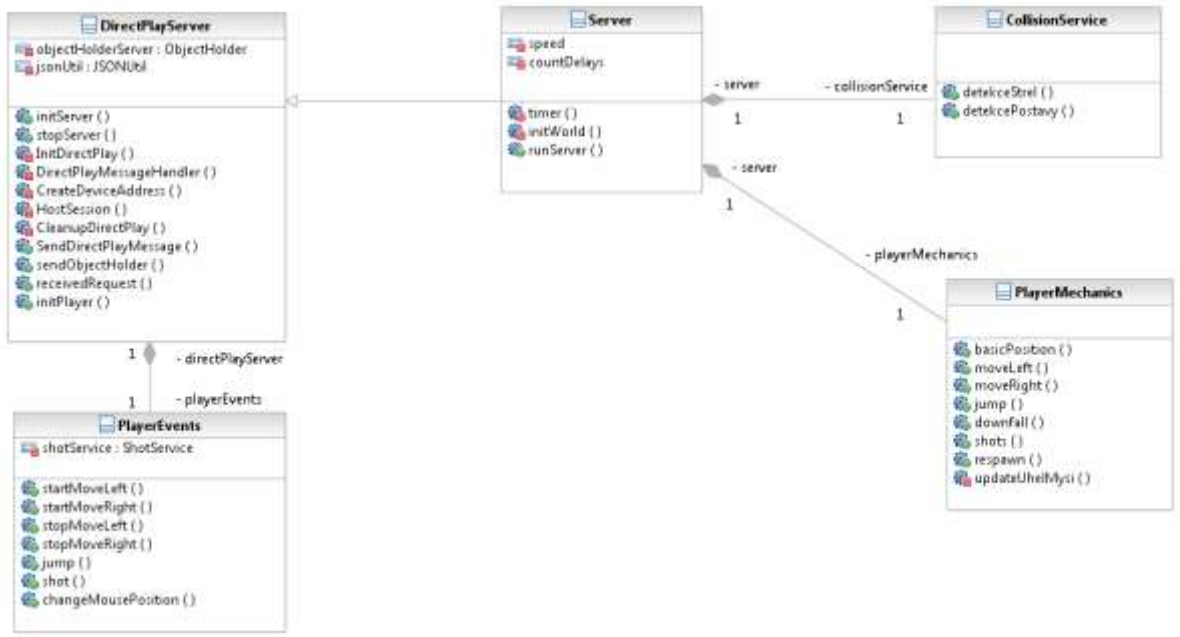
Pro vykreslování hry na obrazovku se používá třída *SkeletAnimation*, která obsluhuje veškeré práce s grafickou knihovnou OpenGL. Obraz je vykreslován s frekvencí 60 snímků za vteřinu. Při každém

vykreslení je načítána aktuální definice světa, která je uložena v proměnné o*bjectHolder*. Tato proměnná je aktualizována s přijatými daty ze serveru (viz. kapitola 6.3.3).

Poslední třídou na straně klienta je *InputsHandler*. Jak z názvu vypovídá, stará se o obsluhu ovládání samotné hry. Stisk klávesy nebo reakce myši jsou zpracovány, a následně odeslány na server. Tímto je zpracování vstupů na straně klienta ukončeno.

6.3.3 Server

Na straně serveru je umístěna hlavní logika hry, která se stará o veškerá pravidla hry a jejich uskutečnění. Jedná se především o samotný pohyb hráče, ale také o vyhodnocování detekce kolize a další funkce. Následující graf popisuje hlavní třídy, které jsou na straně serveru.



Obrázek IX: Třídní diagram – server

Podobně jako na straně klienta, i zde vše zastřešuje třída *DirectPlayServer*, pomocí které se přijímají požadavky a odesílají aktualizované definice světa na všechny klienty. Na tomto diagramu je, stejně jako na předchozím, zobrazena již finální podoba těchto tříd. Zde je také uložena hlavní definice světa v proměnné *objectHolderServer*. V té je uložen aktuální stav na serveru, který má přednost před těmi na klientovi.

Obsluhu odesílání a přijímání zpracovává třída *Server*. Ta v pravidelných intervalech (nastaveno na 30ms) zpracovává pohyb objektů a detekci kolize. Tímto intervalem je tak určena rychlost pohybu (v každé iteraci se pohyb provede vždy o jeden krok).

Veškeré herní mechaniky (animace a pohyby postav a střel) jsou zpracovány ve třídě *PlayerMechanics*. Pohyb hráče je určen pomocí jeho stavů. Z klienta přichází požadavky pouze při změně těchto stavů, a proto není zapotřebí odesílat z klienta informace o stisku klávesy při každé iteraci. Tyto změny stavů hráče, přijatých od klienta, jsou zpracovány ve třídě *PlayerEvents*.

Každému hráči je přiděleno unikátní *id*, které je v každém požadavku odesláno společně s daty. Je tak zaručeno, že akce se provede na správném uživateli.

Po každé iteraci je také prováděna detekce kolize v pomocné třídě *CollisionService*. Po každé změně pohybu se vyhodnotí detekce kolize. Je tak zaručeno, že se tato operace vykoná vždy nad aktuální verzí světa, a je tak spravedlivě vyhodnocena (toto je důležité převážně při detekci kolize střely s hráčem).

6.3.4 Shrnutí

Po rozdělení rolí každé straně je již vše připraveno pro síťovou komunikaci. Je jasné, že všechny prvky nešli bez síťové komunikace nasimulovat, je však již položen základ, ze kterého se dá vycházet při tvorbě síťové komunikace u hry. Tato příprava usnadnila další postup.

6.4 Síťová komunikace

V této chvíli již přišla na řadu nejdůležitější a nejtěžší část celé implementace. Nyní se ukáže, zda příprava byla dostatečná a jaké chyby se při ní vyskytly. Dokončí a shrne se veškerá tvorba hry a mohou se vyvodit závěry z implementace hry.

V následujících podkapitolách jsou shrnuty jednotlivé části přenosu dat po síti. Jedná se o popis využití knihovny DirectPlay, dále formát přenášených dat a popis zpracování na straně klienta a serveru.

6.4.1 DirectPlay

Pro síťovou komunikaci byla zvolena knihovna DirectPlay. Ačkoliv se jedná o knihovnu staršího data (viz. kapitola 3.1.1), jedná se o poměrně kvalitní a výkonnou knihovnu. Tato hra je tedy výhradně určena pro OS MS Windows.

Nemá cenu se zde zabývat ukázkou z implementace. Popis použitých metod je dostupný ve třídních diagramech v kapitolách 6.3.2 a 6.3.3. Ukázky použití jsou dostupné jako součástí DirectX 8.1 SDK. Implementace vycházela z přiložených příkladů a dokumentace, a je přizpůsobena této hře.

Důležité je však zmínit, že veškerá data jsou u této hry přenášena v datovém typu *string* (standardní knihovny C++ std). Následně je tento řetězec převeden na pole bytů a odeslán po síti v datovém packetu (viz. zdrojový kód II). Způsob přenosu dat je popsán v následující kapitole.

6.4.2 Přenos dat

Jak již bylo zmíněno dříve, data jsou kódována na řetězec znaků, a následně přetypována na pole bytů. Poté jsou odeslána přes síť na vzdálené počítače. Data jsou formátována z daných objektů na řetězce pomocí třídy *JSONUtil*, která převádí jednotlivé objekty na textové řetězce (viz. kapitola 6.4.3).

```
JSONNode json = jsonUtil->getObjectHolderJson(objectHolder);
std::string data = json.write();

dpnBuffer.pBufferData = reinterpret_cast<BYTE*>((char*)data.c_str());
dpnBuffer.dwBufferSize = strlen(data.c_str()) + 1;

if( FAILED( hr = g_pDPServer->SendTo(DPNID_ALL_PLAYERS_GROUP,   // dpnid
                                     &dpnBuffer,                // pBufferDesc
                                     1,                         // cBufferDesc
                                     0,                         // dwTimeOut
                                     NULL,                      // pvAsyncContext
                                     NULL,                      // pvAsyncHandle
                                     DPNSEND_SYNC |             // dwFlags
                                     DPNSEND_NOLOOPBACK ) ) )
{
    printf("ERROR: Failed Sending Data:  0x%x\n", hr);
}
```

Zdrojový kód II: DirectPlay - odeslání dat

Na tomto příkladě je ukázka serveru, který odesílá data všem klientům. Veškerá data určená k přenosu jsou uložena v objektu *objectHolderServer* (viz. kapitola 6.3.3) Nejprve jsou všechny objekty serializovány do jednoho JSON objektu, který je reprezentován jako textový řetězec. Ten je následovně převeden na pole bytů a odeslán na klienty. Způsob serializace a deserializace těchto dat je popsán níže.

6.4.3 Formát dat

Formát dat je definován pomocí struktury JSON objektů a polí. JSON je odlehčený formát dat, určený pro přenos dat. Jeho výhodou je lehká čitelnost a jednoduché mechaniky pro generování a parsování dat [10].

Objekty jsou serializovány pomocí JSON nástrojů do textového řetězce. Po přijetí druhou stranou jsou následně deserializovány zpět z řetězce na daný objekt.

6.4.3.1 Definice světa

Následuje ukázka serializace definice světa a jeho jednotlivých objektů. Zde je ukázka převodu objektu hráče do JSON podoby.

```
player: {
      “id”: 12345,
      “HP”: 100,
      “name”: “playerName”,
      “color”: {
            “red”: 1.0,
            “green”: 0.0,
            “blue”: 0.0
      },
      “isPohybVlevo”: true,
      “isPohybVpravo”: false,
      “fazePohybu”: 12,
      “prvniFaze”: false,
      “rozbeh”: false,
      “dobehnutVlevo”: false,
      “dobehnutVpravo”: false,
      “vyskok”: true,
```

```
        "odraz": false,
        "pada": false,
        "mouseX": 200,
        "mouseY": 150,
        "uhelMysi": 15.5,
        "stackX": 0,
        "stackY": 0,
        "bones": [
              {
                    "name": "teloBone",
                    "x": 100,
                    "y": 0,
                    "uhel": 90,
                    "delka": 100,
                    "uhelPosun": 3
              }, { .. } , … // ostatní kosti
        ]
}
```

Jak je vidět na této struktuře, posílá se kompletní definice objektu *Player*, včetně všech jeho kostí. Podobně jsou strukturovány objekty *Shot* a *WorldObject*. Jelikož se jedná o kolekce, jsou odeslány pomocí jednoho JSON objektu, který obsahuje tři pole těchto objektů (viz. struktura níže).

```
{
        "worldObjects": [],
        "shots": [],
        "players": []
}
```

Serializována je tedy kompletní definice světa na serveru. Je tak zaručena maximální objektivita na straně klienta a nemůže dojít k desynchronizaci světa. Nevýhodou však je, že jsou poměrně zbytečně přenášena některá data.

6.4.3.2 Požadavky klienta

Na straně klienta jsou sbírány požadavky, které jsou rovněž strukturovány do formátu JSON objektů. Poté jsou odeslány na server, který je zpracuje. Každý požadavek nese informaci o typu požadavku a identifikaci hráče, kterému daný požadavek patří. Struktura JSON objektu je následující.

```
{
        "type": "changeMousePosition",
        "id": 12345,
        "screen_height": 600,
        "screen_width": 600,
        "x": 263,
        "y": 1
}
```

Toto je ukázka požadavku na změnu polohy myši (zaměřovače). Každý požadavek obsahuje dva povinné atributy *type* a *id* (vyznačeny tučně). *Type* definuje typ požadavku, který se má na straně server vykonat a *id* identifikuje daného hráče. Ostatní parametry jsou přidány vždy podle typu požadavku. V případě změny polohy myši se jedná o *screen_height*, *screen_width*, *x* a *y*.

6.4.4 Klient

Požadavky z klienta na server jsou odesílány vždy při každém vyvolání události (stisk a uvolnění klávesy, klinutí nebo pohyb myši). V daném okamžiku je vytvořen požadavek (viz. kapitola 6.4.3.2), který je okamžitě odeslán na server. Výjimkou je pouze pohyb myši. Ten by generoval velké množství požadavků, a proto je odeslána každá třetí událost myši. Zaměřovač je tak posunut maximálně o tři pixely, což vzhledem k typu hry není takový problém.

Ve vlastním vlákně také běží příjem dat ze serveru. Jakmile se na té části klienta, která sleduje příjem dat ze sítě, objeví data, provede se aktualizace světa. Data se deserializují do objektu *ObjectHolder* (viz. kapitola 6.4.3), který slouží pro uchování dat, využívané při vykreslování. Zde však může nastat problém v případě, kdy jsou data zároveň přepisována nebo čtena. Jedná se o klasický problém sdílení stejných prostředků více vlákny. Z těchto důvodů byly vytvořeny dva zámky.

První se stará zamykání objektu v průběhu čtení (read lock). Tím je zaručeno, že pokud se data načítají (při vykreslování herního světa na obrazovku), nebudou ve stejné chvíli přepsána.

Druhým případem je zámek při editaci dat (write lock). Ten zaručuje naopak to, že data nejsou čtena v případě, kdy jsou aktualizována po přijetí ze serveru.

Tímto je zaručena konzistence jednoho objektu na straně klienta a je ošetřeno, že nedojde k chybě při čtení nebo zápisu dat na jednom sdíleném objektu.

6.4.5 Server

Server se stará o příjem požadavků od všech klientů. Jedná se především o změnu stavu hráče (např. pohyb vlevo, výskok, změna polohy myši, atd.) nebo výstřel (vytvoří se nová instance střely). V pravidelném cyklu jsou paky tyto změny provedeny. Každý cyklus se stará o animaci hráče a pohyb střel. Dále je provedena detekce kolize. Po dokončení všech změn je tato verze světa serializována do řetězce a následně odeslána na klienty.

Zde nedochází k žádnému sdílení prostředků více vlákny, neboť všechny operace probíhají postupně. Z tohoto důvodu není potřeba používat zámky na objekty.

Problém však může nastat v případě, kdy doba vykonávání těchto operací přesáhne dobu určenou pro jednu iteraci (v případě, kdy je tento čas menší se vlákno uspí na potřebnou dobu). Jak se ukázalo, největší problém je právě serializace dat do JSON řetězce. Tato operace v případě hraní více hráčů přesahovala dobu pro jednu iteraci. V této situaci pak nastával problém se zpomalením pohybu. Hra se navíc tímto způsobem neustále zpomalovala. Tento problém vyřešilo jednoduché vynechání této serializace v počtu danou zpožděním na jednu iteraci (např. pro iteraci nastavenou na 30ms při zpoždění 90ms se vynechal tento cyklus 3x). Tím sice nastaly na straně klientů tzv. „lagy“ (záseky ve hře, kdy se najednou provedlo vykreslení o několik kroků). Ačkoliv hra na straně klienta působila trhaně, docílilo se zamezení zpomalení hry.

6.5 Chyby a jejich řešení

Jak se ukázalo, některá rozhodnutí nebyla ideální a na hru měly negativní dopad. Také by bylo možné některé fungující prvky optimalizovat, aby hra dosáhla lepší kvality. Některé z návrhů na zlepšení nebo odstranění chyb zde budou popsány.

Jedná se především o optimalizace při přenosu dat a jejich formátování. Dále by šlo využít některé techniky pro optimalizace řešení zpoždění mezi klientem a serverem.

6.5.1 Optimalizace přenosu dat

Jako poměrně nepovedený se ukázal samotný přenos dat. Ten je jednou z nejdůležitějších prvků této hry, a proto jsou zde popsány návrhy na optimalizace, které lze v budoucnu uskutečnit.

6.5.1.1 Snížení objemu přenášených dat

Největším nedostatkem je především nutnost přenosu kompletní definice světa a hráčů. Vzhledem k neměnnosti definice světa by stačilo odeslat objekty světa pouze jednou, a poté již odesílat pohyblivé objekty (hráče a střely). Hlavní změna by však měla dosáhnout samotná animace. Ta je v současné době stavěna na vykreslování podle definice jednotlivých kostí. Pokud by se však vytvořila animace, která by vykreslovala hráče podle jeho fáze pohybu (proměnná *fazePohybu* ve třídě *Player*), ušetřilo by se velké množství dat nutných pro přenos (především zbytečných definic kostí, které by nebylo nutné odesílat). Takováto optimalizace by ušetřila čas při parsování objektů před odesláním (viz. kapitola 6.4.5).

6.5.1.2 Formát dat

Volba přenosu dat ve formátu JSON objektů také není moc vhodná. Samotný formát je sice čitelný a jednoduchý. Generování objektů však zabírá neúměrné množství času, neboť se všechna data přepisují na řetězce. Tato operace je poměrně pomalá a při generování velkého množství dat značně zdržuje. Ideální by bylo vytvořit mechanismus pro zabalení dat do binárního formátu bez nutnosti tvorby textových řetězců.

6.5.2 Odesílání požadavků

V případě klienta by šlo optimalizovat odesílání požadavků. V této verzi hry se totiž odesílá požadavek na server při každé události. Bylo by však více než vhodné odesílat na server vždy balíček požadavků v nějakém intervalu. V tomto pevně daném intervalu by se posbíraly všechny požadavky, odstranily by se zastaralé (např. při pohybu myši by se odeslal pouze jeden požadavek na nejaktuálnější pozici myši). Následně by se v jednom datovém balíčku odeslaly na server. Ušetřila by se tak nejen datová linka, ale také zátěž při zpracování požadavků na straně serveru.

6.5.3 Další možná vylepšení

Po vyřešení předchozích problémů by se dalo uvažovat nad dalšími optimalizacemi. Jednou z nich je například implementace interpolace. V případě, že by u klienta nedošlo k aktualizaci světa v dané iteraci (což se v současné době stává poměrně často a zapříčiňuje tak neplynulost hraní), předvídal by se stav světa podle pravidel hry a stavu předchozích verzí (viz. kapitola 5.3.1).

Dalším možným vylepšením by mohlo být předvídání vstupů. Na straně klienta je možné podle známých pravidel hry předvídat pohyb hráče, který by se na straně klienta pohyboval bez znatelného zpoždění (viz. kapitola 5.3.2). Byla by však nutná kontrola polohy daného hráče mezi verzí na straně klienta a serveru. V první řadě by tedy bylo nutné zavést synchronizační hodiny.

6.6 Shrnutí a závěr

Ačkoliv hra obsahuje ještě řadu chyb, byla vytvořena jednoduchá hra, která demonstruje hru pro více hráčů. Při vytváření hry došlo k několika chybám, které zapříčiňují nedostatečnou plynulost hry se špatným připojením nebo při hraní více hráčů. Základy hry jsou však položeny velmi dobře, a dá se na nich v budoucnosti postavit kvalitní síťová hra.

Také samotná hratelnost není úplná. V současné době se jedná pouze o prototyp hry, nad kterým lze postavit plnohodnotnou hru. Je však nutné optimalizovat například detekci kolize, která ještě v některých momentech hry nevyhodnocuje kolizní pozice zcela správně.

7 Závěr

Cílem této práce bylo popsat síťovou komunikaci u počítačových her. V první části tohoto textu tak byli popsány a porovnány síťové knihovny, použitelné pro počítačové hry. Provedla se řada testů a podrobně se jednotlivé knihovny prozkoumaly. Jednou z hlavních zkoumaných knihoven byla DirectPlay, která byla využita při implementaci jednoduché síťové hry.

Při porovnání těchto knihoven vyšla jako pomyslný vítěz knihovna SyncSys. Dosáhla jedny z nejlepších výsledků při testování zpoždění a její zpracování je také na dobré úrovni. Ostatní knihovny buď neobstáli v těchto testech, nebo nebyly příliš kvalitně zpracovány. Nejlepších výsledků se však dosáhlo při využití síťového rozhraní WinSock, který je součástí knihoven Windows. Otázkou tak zůstává, zda není lepší využít tohoto prostředku pro síťovou komunikaci, případně si vytvořit vlastní síťové rozhraní, postavené na této technologii.

Dále v tomto textu byly popsány některé přístupy při tvorbě síťových her. Jedná se především o rozdělení autority mezi klienty a server. Dále bylo rozvedeno širší pojednání o způsobu, kdy se autoritou stává server, a klient slouží pouze pro ovládání hry a zobrazování na výstupní zařízení. Tento způsob implementace, jeho problémy a jejich možná řešení byly v teoretické rovině podrobněji vysvětleny.

V poslední části byla implementována jednoduchá síťová hra pro více hráčů, vytvářena na základě předchozí teorie. Nebylo účelem vytvořit kompletní dílo, ale vytvořit prototyp síťové hry, na kterém se dá demonstrovat ukázka přístupu k síťové komunikaci. Tato hra byla vytvářena pomocí iterací. Nejprve se vytvořila grafická stránka hry, následně herní mechaniky, rozdělení na stranu klienta a serveru a v poslední fázi byla vytvořena samotná síťová komunikace. Pomocí tohoto přístupu byla vytvořena funkční hra pro více hráčů.

Tato hra však obsahuje ještě několik drobných chyb, převážně v části síťové komunikace. Ty jsou zdokumentovány a je možné je v budoucnu upravit. Je však vytvořen velmi kvalitní základ, na kterém se dá postavit kvalitní síťová hra bez nutnosti velkých zásahů do zdrojových kódů.

Všechny předem zadané cíle tak byly splněny. Práce však, především na samotné implementaci hry, odevzdáním této práce nekončí. Plánují se odladit všechny vzniklé chyby, případně přidat různé optimalizace u síťové komunikace. Také grafická stránka hry a samotná hratelnost může být upravena do kvalitnější podoby.

8 Literatura

[1] *RakNet* [online]. 2011 [cit. 2011-04-13]. RakNet - Multiplayer game engine. Dostupné z WWW: <*http://www.jenkinssoftware.com/*>.

[2] *OpenTNL* [online]. 2011 [cit. 2011-04-13]. Torque Network Library. Dostupné z WWW: <*http://www.opentnl.org/*>.

[3] *Hawk Software* [online]. 2004 [cit. 2011-04-13]. HawkNL (Hawk Network Library). Dostupné z WWW: <*http://hawksoft.com/hawknl/*>.

[4] *ENet* [online]. 2011 [cit. 2011-04-13]. ENet. Dostupné z WWW: <*http://enet.bespin.org/*>.

[5] *Zoidcom* [online]. 2011 [cit. 2011-04-13]. Zoidcom Network Library. Dostupné z WWW: <*http://www.zoidcom.com/*>.

[6] *SyncSys* [online]. 2011 [cit. 2011-04-13]. SyncSys networklib. Dostupné z WWW: <*http://syncsys.sourceforge.net/*>.

[7] GRYGÁREK, Petr. *Počítačové sítě*. VŠB – TU Ostrava : 2005 [cit. 2011-04-14]. Počítačové sítě (POS) Skripta (2004-2005)

[8] *Source Multiplayer Networking* [online]. 2011 [cit. 2011-04-13]. Source Multiplayer Networking – Valve Developer Community. Dostupné z WWW: <*http://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking/*>.

[9] *LogMeIn* [online]. 2011 [cit. 2011-04-13]. LogMeIn – Remote Access and Desktop Control Software. Dostupné z WWW: <*https://secure.logmein.com/*>.

[10] *JSON* [online]. 2011 [cit. 2011-04-13]. Introducing JSON. Dostupné z WWW: <*http://www.json.org/*>.

[11] *libjson* [online]. 2011 [cit. 2011-04-13]. Libjson. Dostupné z WWW: <*http://libjson.sourceforge.net/*>.

Přílohy

A Uživatelská dokumentace

1 Instalace aplikace

Pro otevření zdrojových souborů je zapotřebí využít vývojového nástroje MS Visual Studio 2010 (a novější). Zpětná kompatibilita není ověřena. Po otevření projektu je zapotřebí přilinkovat hlavičkové soubory a potřebné knihovny.

Statické knihovny jsou umístěny v adresáři *src/include/lib/*

Hlavičkové soubory jsou umístěny v adresáři *src/include/headers/*

Dynamické knihovny jsou umístěny v adresáři *src/include/dll/*

Na každém projektu je zapotřebí zvolit adresář pro přidané knihovny vždy, podle daného umístění na pevném disku (Configuration Properties -> Linker -> General -> Additional Library Directories).

Dále je nutné zvolit adresář pro hlavičkové soubory obdobně, jako v předchozím případě (Configuration Properties -> C/C++ -> General -> Additional Include Directories).

2 Manuál

Zde jsou popsány základní ovládací prvky hry (spuštění hry a ovládání).

2.1 Spuštění hry

Hra se spouští pomocí konzole. Každá operace je podrobně popsána. Pro ukončení hry postačí zavřít okno s právě hranou hrou.

Po spuštění aplikace se zobrazí následující dotaz:

Write command 'client' or 'c' for CLIENT, 'server' or 's' for SERVER:

Pro start serveru stačí zadat příkaz „**s**“ případně „**server**“ a potvrdit stiskem klávesy „**Enter**“. Pro start klienta stačí zadat příkazy „**c**“ nebo „**client**“.

2.1.1 Server

Po spuštění serveru je spuštěna vývojová konzole, kde se zapisují zápisy do logu. Pro ukončení serveru stačí uzavřít toto okno.

2.1.2 Klient

Po startu klienta se objeví instrukce pro zadání informací o uživateli.

Write your name:

Po zobrazení tohoto příkazu zadejte své jméno, pod kterým budete reprezentováni ve hře.

Choose color:

Z nabídky barev vyberte jednu podle zadaného čísla. Pokud zadáte jiné číslo, než je na výběr, bude Vám přidělena základní bílá barva.

Please enter the IP address of the host:

Zde je nutné zadat IP adresu, na kterém běží server (viz. 2.1.1). Je nutné, aby byl nejprve spuštěný server, poté se lze přihlásit na tento server pod danou IP adresou.

Po úspěšném zadání všech informací budete přihlášení k serveru a hra započne.

2.2 Ovládání

Hra se ovládá pomocí klávesnice a myši.

Klávesy:

- „A" – pohyb vpravo
- „D" – pohyb vlevo
- „mezerník" – výskok

Myš:

- „Levé tlačítko" - výstřel

B Obsah CD

Adresář	Popis
/src/app	Zdrojové soubory multiplayerové hry
/src/testy	Zdrojové soubory testů
/src/include/lib	Knihovny
/src/include/headers	Hlavičkové soubory
/doc/text	Text diplomové práce
/doc/manual	Uživatelská příručka